

AD-A072 140

ARMY AVIATION RESEARCH AND DEVELOPMENT COMMAND ST LO--ETC F/G 9/2
SCHEDULE. MANAGERIAL TOOL FOR PROJECT/PRODUCT MANAGERS.(U)
MAY 79 V ALLEN, L ANTWILER, M GARSIK

UNCLASSIFIED

USAAVRADCOM-TR-79-17

NL

1 of 3

AD
A072140



USAAVRADCOM
TR 79-17

LEVEL

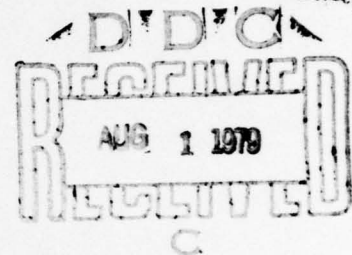
(12)
B.S.

SCHEDULE

AD A 072140

Managerial Tool For Project/Product Managers

Dr. Vernon Allen
Mr. Lonnie Antwiler
Mr. Michael Garsik



MAY 1979

DISCLAIMER STATEMENT

THE VIEW, OPINIONS, AND/OR FINDINGS CONTAINED
IN THIS REPORT ARE THOSE OF THE AUTHOR(S)
AND SHOULD NOT BE CONSTRUED AS AN OFFICIAL
DEPARTMENT OF THE ARMY POSITION, POLICY, OR
DECISION, UNLESS SO DESIGNATED BY OTHER
DOCUMENTATION.

FINAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

FILE COPY

US ARMY AVIATION RESEARCH AND DEVELOPMENT COMMAND
Directorate for Development and Engineering
Scientific and Engineering Computational Office
P.O. Box 209
St. Louis, Missouri 63166

79 07 30 069

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER USAAVRADCOM TR 79-17	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SCHEDULE Managerial Tool For Project/Product Managers	5. TYPE OF REPORT & PERIOD COVERED Final Report, May 1979	6. PERFORMING ORG. REPORT NUMBER USAAVRADCOM-TR-79-17
7. AUTHOR(s) Dr. Vernon Allen Dr. Lonnie Antwiler Dr. Michael Garsik	8. CONTRACT OR GRANT NUMBER(s)	
9. PERFORMING ORGANIZATION NAME AND ADDRESS USAAVRADCOM, DRDAV-ES PO Box 209 St. Louis, MO 63166	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	
11. CONTROLLING OFFICE NAME AND ADDRESS USAAVRADCOM, DRDAV-ES PO Box 209 St. Louis, MO 63166	12. REPORT DATE May 1979	13. NUMBER OF PAGES 190
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 12 194 P	15. SECURITY CLASS. (of this report) UNCLASSIFIED	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES This effort was funded by the Aircraft Survivability Equipment (ASE) Project Manager's Office, St. Louis, MO.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Schedule, Milestone, Graphics, Managerial Tool, Tracking		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Schedule is an interactive computer program designed to organize and present schedule information. It enables the user to maintain a master file of schedule type information for each of many projects, from which current status, change impact analysis or "what if" studies can be obtained. The program, which was developed on an IBM 360/65, provides graphic output on a Tektronix 4014 or tabular output on most printers/typewriters. It was written in PL/1 language except for the graphics routines, which are combinations of Fortran IV and Tektronix Advanced Graphics, Release 3.		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

USAAVRADCOM TR 79-17

SCHEDULE

Managerial Tool For Project/Product Managers

Dr. Vernon Allen
Mr. Lonnie Antwiler
Mr. Michael Garsik

MAY 1979

FINAL REPORT

Approved for Public Release; Distribution Unlimited

US Army Aviation Research and Development Command
Directorate for Development and Engineering
Scientific and Engineering Computational Office
PO Box 209
St. Louis, Missouri 63166

FORWARD

SCHEDULE was written by Dr. Vernon Allen. Graphics routines and interactive features were provided by Mr. Lonnie Antwiler. The documentation was written by Mr. Michael Garsik. The three authors are all employed by the US Army Aviation Research and Development Command (AVRADCOM).

Funds were provided by the Aircraft Survivability Equipment (ASE) Project Manager's Office, St. Louis, MO.

Table of Contents

	Page
1.0 Introduction	
1.1 Abstract	1
1.2 Language	1
1.3 Hardware	1
1.4 Overlay Structure	1
1.5 Files	1
1.6 Data Types	2
1.7 Number of Subroutines	2
1.8 Graphics	2
2.0 SCHEDULE Program	
2.1 Introduction	3
2.2 Subroutine ADDD	3
2.3 Subroutine CHANGE	3
2.4 Subroutine FORM	3
2.5 Subroutine LLIST	3
2.6 Subroutine MASTER	4
2.7 Subroutine NEW1	4
2.8 Subroutine NEW	5
2.9 Subroutine REPORTS	5
2.10 Subroutine SSLIP	5
2.11 Subroutine TYREPT	5
2.12 Subroutine UPDATE	5
2.13 Subroutine WINDOW	6
2.14 Minor Subroutines	6

	Page
3.0 Programmers Guide	
3.1 Introduction	6
3.2 SCHEDULE Setup	7
3.3 User Setup	7
3.4 Maintenance	8
3.5 Standard Titles	8
3.6 Change Recording	9
 4.0 User's Guide	
4.1 Introduction	9
4.2 Entering New System	10
4.2.1 Entering New System From a Keyboard	10
4.2.2 Entering New System From a File	17
4.2.2.1 Report Form	19
4.2.2.2 Non-Report Form	22
4.3 Editing Old Systems	23
4.3.1 General	23
4.3.2 Add	25
4.3.3 Delete	29
4.3.4 List	30
4.3.5 Change	35
4.3.5.1 Development Step Record Changes	35
4.3.5.1.1 Title	36
4.3.5.1.2 Data	37
4.3.5.1.3 Application Code	38
4.3.5.1.4 Element Code	39

	Page
4.3.5.2 Remarks Record Changes	40
4.3.5.2.1 Remark	40
4.3.5.2.2 Application Code	41
4.3.5.3 Header Information Changes	42
4.3.5.3.1 Aircraft System	43
4.3.5.3.2 System Title	43
4.3.5.3.3 As Of Date	44
4.3.5.3.4 Start Year of Report	45
4.3.5.3.5 Number of Years in Report	46
4.3.5.3.6 Application Code	47
4.3.5.4 Return	47
4.3.6 Slip	48
4.4 Write Reports	52
4.4.1 Graphic Reports	52
4.4.2 Typewriter Reports	58
4.5 Special Options	62
4.5.1 General	62
4.5.2 Window	62
4.5.3 Lines Per Page	63
4.5.4 Application Code Prompting List	65
4.5.5 Cross System	66

	Page
APPENDIX A - SCHEDULE Subroutine Listings	A-1
Subroutine ADDD	A-2
Subroutine CHANGE	A-7
Subroutine FDATE	A-15
Subroutine GRAP	A-17
Subroutine LLIST	A-18
Subroutine MASTER	A-21
Subroutine NEW	A-24
Subroutine NEW1	A-31
Subroutine RDATE	A-36
Subroutine REPORTS	A-37
Subroutine SSLIP	A-46
Subroutine TYREPT	A-49
Subroutine UPDATE	A-52
Subroutine WINDOW	A-62
APPENDIX B - Overlay Listing	B-1
APPENDIX C - Maintenance Subroutine Listings	C-1

List of Figures

Figure		Page
1.1	Single Region Overlay Tree Structure	1A
3.1	SCHEDULE Subroutines	7A
3.2	SCHEDULE.CLIST Listing	7B
3.3	SCHEDULE Passed Parameter Values	7C
3.4	Sample Standard Title Dataset Listing	8A
4.1	Full/No Prompting Example	10A
4.2	Sample Single and Multiple Event Input	15A
4.3	Sample Graphic Report	25A
4.4	Sample Graphic Report Record Listing	25B
4.5	Sample Single and Multiple Task Input	27A
4.6	System Listing with New Record Added	29A
4.7	Sample System Listing	33A
4.8	Sample Output Listing	33B
4.9	Sample Output Listing	34A
4.10	Sample Report (Not Slipped)	49A
4.11	Sample Report (Slipped)	51A
4.12	Sample Graphics Report	55A
4.13	Sample Typewriter Report	61A
4.14	Sample Report (No Window)	64A
4.15	Sample Report (Windowed)	64B
4.16	Lines Per Page Off	65A
4.17	Lines Per Page On	65C

1.0 INTRODUCTION

1.1 ABSTRACT

SCHEDULE is an interactive computer program which serves as a management tool by helping to organize project schedule information. It enables the manager to maintain a master file of schedule information on one or many projects from which reports may be prepared that display the most recently updated schedule status. The creative user can use SCHEDULE to "play" with the data and prepare alternative schedules showing the impact of delays or changes on a project.

1.2 LANGUAGE

With the exception of one subroutine SCHEDULE is written in PL/I. The exception to this is the graphics portion of the program which is a combination of FORTRAN IV and TEKTRONIX Advanced Graphics, Release 3. Any later release of the TEKTRONIX graphics package will also be suitable for the operation of SCHEDULE.

1.3 HARDWARE

SCHEDULE was developed and debugged using an IBM 360/65 Computer System located at the U.S. Army Troop Support and Aviation Materiel Readiness Command in St. Louis, MO. The program was designed to provide a graphic report on a TEKTRONIX 4014 Computer Display Terminal or a tabular report on most types of printer or typewriter TSO terminal, such as the Texas Instruments Silent 700 or a Digital DECWRITER II. Schedules may be edited from any kind of TSO terminal.

1.4 OVERLAY STRUCTURE

Because of the amount of storage required to load both the program and the data into core, it became necessary to design an overlay structure for SCHEDULE to facilitate the execution of the program. Figure 1.1 is the single-region tree that is used to structure SCHEDULE into an overlay. Subroutine OVR.CLIST will create this overlay structure when executed in conjunction with OVR.OBJ which it references. How to do this is outlined in Section 3.2.

1.5 FILES

To allow a program such as SCHEDULE to deal primarily with the logical aspects of data rather than with its physical organization in a data set, PL/I employs a symbolic representation

of a data set called a file. SCHEDULE employs two files named INDEX and DATA.DATA.

INDEX is used to permanently store schedule data and is used for no other purpose; this is done in order to decrease the chance of data being lost when SCHEDULE is being executed. Data to be edited or reported on is read into the buffer file, DATA.DATA, where all data manipulation takes place; data which has been edited is then rewritten back into INDEX for permanent retention. All data transmission is record oriented.

1.6 DATA TYPES

There are four different kinds of records, each with its own format. The record types and the data included in each are:

- a. Header Record - Major Title, Subtitle, As-Of-Date, Start Year, and Number Of Years In Report.
- b. Task Record - Standard Title I.D. (if used, otherwise blank), Application Code, Task Title, and up to three Stop Date/Start Date/Percent Completion combinations.
- c. Event Record - Standard Title I.D. Number (if used, otherwise blank), Application Code, Task Title, and up to nine separate Event Dates.
- d. Remark Record - Application Code, Remark Text.

1.7 NUMBER OF SUBROUTINES

SCHEDULE is made up of seventeen subroutines that are arranged in an overlay structure as previously discussed. Listings of all the subroutine which make up SCHEDULE are included in Appendix A. There are also two other groups of programs which are included in the SCHEDULE package whose functions are not directly related to the execution of the program but without which it could not be run. The first of these is required to setup the program on a computing system and to allow users to set themselves up so that they may use SCHEDULE. The second group is necessary to maintain the data files in order to save space in the memory of the machine and to aid in the trouble-shooting of the files should problems develop. Both groups of programs and their use are covered in Section 3.0

1.8 SCHEDULE GRAPHICS

The graphics routine FORM.FORT was written by Mr. Lonnie Antwiler using FORTRAN IV and TEKTRONIX Advanced Graphics Release 3.0 routines. Complete documentation of this routine is currently

being prepared for future publication, hence a complete discussion of it will not be included here.

2.0 SCHEDULE PROGRAM

2.1 INTRODUCTION

As mentioned, SCHEDULE is made up of 17 subroutines of which 12 are major, that is, they perform a task or tasks which are user directed or controlled. The remaining 5 subroutines are minor, performing tasks that are, in most cases, transparent to the user who has no control over them. Herein are descriptions of each of the major subroutines covering their basic purpose and functions, followed by a brief look at the minor programs.

2.2 Subroutine ADDD

When a new record is to be added to a system, ADDD is called to prompt the user for the new information. ADDD will display the proper formats to be followed and write the new data into the buffer file as it is entered. After the complete record has been inputted, control returns to the editing routine UPDATE, note that the data is not written into the permanent storage file at this time.

2.3 Subroutine CHANGE

This subroutine allows users to change a part of an existing record that has been copied into core from an earlier step that specified what record was to be changed. CHANGE will display the current data that is stored in the record and will prompt the user by displaying the proper format for the new data. After the change has been entered, control will be returned to the editing routine UPDATE, note again that the new data is not entered into the permanent storage at this time.

2.4 Subroutine FORM

This subroutine is completely responsible for all of the graphics required by SCHEDULE. It is called when a graphic report is to be displayed on a TEKTRONIX 4014 and generates the form for the report along with the appropriate data symbology. Upon completing the report, control is returned to subroutine MASTER.

2.5 Subroutine LLIST

Also part of the editing portion of SCHEDULE, this subroutine will provide the user with either a listing of all the records

contained within a system or with a listing of the heading information for all the systems contained in the permanent storage file. This listing may be obtained during the TSO session and displays the application code and record code number when listing records. After the list is complete control will be returned to the editing routine UPDATE.

2.6 Subroutine MASTER

This subroutine is the initial entry point to SCHEDULE and acts as the main procedure as required in PL/I. This subroutine is unique in that it receives arguments passed from the clist that executes SCHEDULE (How to initialize these arguments is covered in Section 3.3). These arguments enable or disable the currently available special options: Application Code Prompting List, Cross System Reporting, Lines Per Page and Windowing.* The Position Dependent Application Code is not available at the present time. Briefly, these options do the following:

Application Code Prompting List - allows the user to display a specific list of application codes to be used as a prompt to the request for the codes.

Cross System Reporting - allows the user to prepare a report on all systems at one time rather than on a system-by-system basis.

Lines Per Page - allows the user to change the default value of thirty lines maximum per report page, to a value ranging from nine to thirty.

Windowing - allows the user to display, in a report system, SCHEDULE data from a specified period of time.

MASTER is also responsible for determining the task to be accomplished and routing control to the correct portion of SCHEDULE, as well as sending control back to the user when the session is concluded.

2.7 Subroutine NEW1

NEW1 is called when a user decides to enter a new system from the keyboard. Before proceeding into the input phase, NEW1 will ask if an old job of inputting data is being restarted and if so, will search the buffer file to determine if enough data is present to pick up from the last record entered. If that is the case, the last record entered will be displayed and the user will be prompted to continue inputting data. If enough data is not present, however, or if an old job is not being restarted, the subroutine will prompt the user for header data, development step data, and remark data if any. Regardless of what path is followed, the data will only have been written out to the buffer file when

*These options are discussed in detail in the Special Options section of the User's guide (Section 4.5).

the input is complete and control is returned to the input subroutine NEW.

2.8 Subroutine NEW

This subroutine initiates the input portion of SCHEDULE by determining from the user where the new data is to be entered from either a file or the keyboard. In the case of entering data from the keyboard, control is sent to NEW1 and returns when the new system is stored in the buffer file. If the data to be entered is already in the buffer file, NEW goes directly to it for the information. In any event, when the new data is ready, this subroutine writes it out to the permanent storage file when instructed to do so and assigns a system code to the new system. Control is then returned to subroutine MASTER.

2.9 Subroutine REPORTS

REPORTS is designed to provide an interface between the language and graphics portions of SCHEDULE. It combines format and title information supplied by the user, along with system data taken from the permanent storage file together in the buffer file in such a way that it is understood by the graphics subroutine FORM.

2.10 Subroutine SSLIP

SSLIP is called when the user decides to slip (i.e., move back in time) any of the tasks or events that make up the schedule data of a system. Part or all of the data may be moved back based on the respective application code of the individual records or tasks may be extended beyond their original ending date. When the "slip" is complete control is returned to the editing routine.

2.11 Subroutine TYREPT

This subroutine is called when a user decides to print out a report on any terminal other than a TEKTRONIX 4014. It reads data from the buffer file which has been stored there by subroutine REPORTS and prints it out in tabular form. As with the graphic report it may consist of individual reports on any number of the systems in the permanent file or of a single report covering all the systems. When the report is complete, control is returned to subroutine MASTER.

2.12 Subroutine UPDATE

All editing done to individual records is controlled through

this subroutine. UPDATE will determine from the user the record to be changed and the type of change to be made to it. It then proceeds to call up the appropriate editing routine, either ADDD, CHANGE, or LLIST, to which it will pass this information. When the editing is complete, UPDATE will return control to subroutine MASTER.

2.13 Subroutine WINDOW

When the window option has been enabled and requested, this subroutine will be called up to determine from the user the upper and lower (start and stop) dates of the window. This information is passed to REPORTS when the system data is being prepared for reporting.

2.14 Minor Subroutines

The five minor subroutines and their functions are:

- a. Subroutine FDATE - Converts a day-month-year formatted response to a numeric calendar code used internally by SCHEDULE.
- b. Subroutine GRAP - Employed by MASTER to route control to the proper subroutine based upon what the user has requested to do.
- c. Subroutine OCFIL - Opens the permanent data file INDEX in such a way that is compatible with the direction of the data transmission that is to take place.
- d. Subroutine RDATE - Converts the numeric calendar code used internally by SCHEDULE to a day-month-year format for display on the terminal.
- e. Subroutine REVERSE - Used to properly position the cursor on a CRT terminal over the format prompts displayed by SCHEDULE.

3.0 PROGRAMMERS GUIDE

3.1 Introduction

This section is intended to assist in preparing SCHEDULE for operation and setting up the catalogs of new users such that they may execute the program. The following instructions can be carried out once SCHEDULE and all of its associated routines have been successfully copied into a users catalog or a catalog of its own. It will be necessary to change any fully qualified dataset names in the clists or routines to reflect the new catalog name. Attempts should be made not to change any of the dataset names, however, if this is not possible care should be taken to see that

all references to that dataset are also changed accordingly.

3.2 SCHEDULE SETUP

Complete the following steps in sequence:

- a. Compile the subroutines listed in Figure 3.1 and save the resulting object modules.
- b. Create a partitioned data set named MODLIB.OBJ and copy the following object modules into it assigning the member names shown in parenthesis after the module names: OCFILE.OBJ (OC), REVERSE.OBJ (R), RDATE.OBJ (RDT), and FDATE.OBJ (DT).
- c. Allocate in the same catalog in which SCHEDULE will be stored two files named STDTIT.DATA and ACHGN (see sections 3.5 and 3.6).
- d. Execute OVR.CLIST to create the overlay structure required by SCHEDULE.

When the overlay has been created, the result will be a load module named MASTER.LOAD which contains the overlay of SCHEDULE. This is the load module that is called by all users of SCHEDULE.

3.3 USER SETUP

Prior to setting up SCHEDULE users, it will be necessary for the programmer to initialize the passed parameters in the SCHEDULE.CLIST dataset, which is copied onto all the users catalogs during the setup procedure. As previously discussed in Section 2.6, these parameters are passed to SCHEDULE and enable or disable the special options that are available. Figure 3.2 is a listing of SCHEDULE.CLIST that shows the parameter format and Figure 3.3 is a list of possible parameter values. Note from Figure 3.2 that regardless of the initial parameter values, the user may override them through the use of positional parameters.

The SCHEDULE support routines are written to allow any user to set themselves up simply by executing SETUP.CLIST, which is part of the documentation package. This routine will create in the users file, the buffer dataset DATA.DATA and the permanent dataset INDEX, both of which are required by the program. A background job is also submitted by the setup routine to initialize INDEX and enter some sample data consisting of one system SCHEDULE into it. Finally, the CLIST required by the user to execute the program, SCHEDULE.CLIST, is copied into their catalog. This completes the setup procedure, most of which the user will be unaware of, however, there will be a dataset named JBA.CNTL present in the user's catalog that is a by-product of the setup process. This dataset is not required by SCHEDULE and in the interest of saving space should be deleted. When all the data sets required by

SCHEDULE are present, the user is then able to begin using the program, hopefully after reading the User's Guide in Section 4.0.

3.4 MAINTENANCE PROGRAM

These routines are provided to permit the user to perform some basic "maintenance" on the permanent master data file, INDEX, that can prevent or solve problems which might occur during the routine use of SCHEDULE. Each routine performs a particular task, however, they all operate in basically the same way. When one of these routines is called, it edits a dataset named COMPRS.CNTL, which contains a general list of Job Control Language that employs an IBM Utility Program, IEBISAM. The editing consists of adding, deleting, or changing lines in COMPRS.CNTL and saving the result in a newly created temporary dataset. This revised control listing is then submitted as a background job.

Listings of all the routines are contained in Appendix C and it is suggested that before attempting to use any of them that the user become familiar with the IBM Utility Program, IEBISAM. Following are brief descriptions of the tasks performed by each of the routines:

- a. COMPRS - Copies INDEX from one direct access volume to another and back to the original volume. During this process deleted records are removed and records in the overflow area of the original dataset are moved onto the primary area.
- b. DUMP - Copies INDEX to tape from disk.
- c. LIST - Copies INDEX and routes the output to a line printer.
- d. RESTORE - Copies tape created by DUMP into INDEX.

3.5 STANDARD TITLE

The purpose of Standard Titles is to allow an office or organization to employ a standard set of titles to describe the various tasks and events that make up the system SCHEDULES. When a new system is being entered from the keyboard, the user is given the chance of using a standard title or entering one of their own. The standard titles are stored in the dataset, STDTIT.DATA, which should be kept in a general catalog available to all users of SCHEDULE. This dataset is allocated to the user when the program is executed and accessed when a standard title is requested. The file itself should be structured using fixed blocking, along with a logical record length and blocksize of 80 characters. A sample listing of a standard title dataset is shown in Figure 3.4. If the Standard Title option is to be used, it will of course, be necessary to place the titles in this dataset, otherwise, this file may be left empty without affecting the operation of the

program as long as Standard titles are not requested by the user.

3.6 CHANGE RECORDING

The ability to record significant changes made to different system SCHEDULES was requested by users to enable them to list from a separate file, all the major changes rather than having to look for them system-by-system. This option is currently not included as part of SCHEDULE, however, the dataset in which these changes would be stored is required for the operation of the program. Similar to the standard title dataset, a dataset named ACHGN must be set up in the general catalog in which SCHEDULE is to be kept so that it can be accessed by all of the users. ACHGN should be allocated using fixed blocking and a logical record length of 114 characters and a blocking size of 1140 characters.

4.0 USER'S GUIDE

4.1 INTRODUCTION

This section is intended to familiarize the new user with all the options available under SCHEDULE and how to use them. Each example introduces a new feature and presents it as a separate task, that is, it begins from a READY mode and concludes with a READY mode. This should not be taken to mean that it is necessary to leave the main program each time a different task is required, the fact is that several opportunities are given to start a new task prior to leaving the program. The reasons for presenting the material in this manner are to illustrate each feature in a straight-forward manner and avoid confusing the user by attempting to present several concepts at once. Points to keep in mind when covering this material are:

- a. Whenever "none" is available as a response to a question its selection will result in control being returned to the next higher level.
- b. The SCHEDULE prompts shown are accurate in their content, however, the format may be slightly different when displayed on the screen.
- c. The "lost time" spent reading this material will be more than recovered by the improved performance of the first-time user who will have a better grasp of what is required by the system and how the data can be prepared to facilitate the task of entering it.

4.2 ENTERING A NEW SYSTEM

4.2.1 ENTERING A NEW SYSTEM FROM THE KEYBOARD

From the READY mode, call up SCHEDULE as shown:

```
READY
EXEC SCHEDULE
```

The screen will erase and SCHEDULE will inquire:

```
DO YOU NEED FULL PROMPTING?
```

to which a yes/no reply is required. A yes response will cause acceptable responses to be displayed after multiple choice questions and a no response will suppress this. An example of both cases is shown in Figure 4.1. Full prompting will be used for this example, so the response to this question is yes (Y):

```
DO YOU NEED FULL PROMPTING?
Y
```

SCHEDULE will then ask for the system type;

```
ENTER SYSTEM TYPE
N - NEW (THE SYSTEM DOES NOT HAVE A FILE CODE)
O - OLD (THE SYSTEM IS IN THE MAIN FILE)
```

which must be responded to with "N" when a new system is to be entered:

```
ENTER SYSTEM TYPE
N - NEW (THE SYSTEM DOES NOT HAVE A FILE CODE)
O - OLD (THE SYSTEM IS IN THE MAIN FILE)
N
```

At this point SCHEDULE must be told where the new data is to be entered from:

WILL YOU ENTER THE DATA FROM THE KEYBOARD OR
IS IT PRESENTLY ON A FILE?

ENTER

K - KEYBOARD

F - FILE

N - NONE

Entering data from a file involves some additional steps beyond this example and is covered in Section 4.2.2. For this case, assume that data is to be entered from the keyboard (which by the way will be the rule rather than the exception) and indicate this by entering "K":

WILL YOU ENTER THE DATA FROM THE KEYBOARD OR
IS IT PRESENTLY ON A FILE?

ENTER

K - KEYBOARD

F - FILE

N - NONE

K

SCHEDULE then asks:

IS THIS A RE-START OF AN OLD JOB?

This may seem like an odd question, yet it offers the user the opportunity to recover seemingly lost data. When a new system is originally being entered, it is first stored in an intermediate storage file separate from the permanent master storage file. Only after the input is complete is the intermediate storage copied into the master file. Unfortunately, if the system should go down for some reason while the data is being entered, the information contained in the intermediate storage file will never get to the permanent main file.

This does not necessarily mean that the data entered to that point is lost, since it may be recoverable from the intermediate storage file. A yes response to this question will cause a search to be made of it to see if there is data present and if it is of sufficient quantity, to allow the user to pick up from where they left off or slightly before that point. If this is the case, SCHEDULE will display the last record saved and proceed to prompt for the next required entry.

If not, this will be indicated and the data input must be started from the beginning. Of course, for the first run through a new system the answer to this question should be "No" (N).

IS THIS A RE-START OF AN OLD JOB?
N

At this point SCHEDULE will enter the input mode which begins with a prompt for the system title followed by several requests for additional descriptive information:

ENTER MAJOR SYSTEMS
/-----/*

This title becomes part of the master storage file and will be displayed on any report (with the exception of a cross system report discussed later) that lists all or part of the information contained in this system. For this example:

ENTER MAJOR SYSTEMS
/AIRCRAFT-----/SUPPORT SYSTEM-----/

The next prompt is for a project title:

ENTER PROJECT TITLE
/-----/

which also becomes part of the permanent master storage file under the major systems. Note in the example response it is not necessary to enter a character(s) in both sections of the prompting format:

ENTER PROJECT TITLE
/GROUND SUPPORT UNIT----/-----/

SCHEDULE will next ask for:

AS OF DATE
/--*---*/

The answer to this should not necessarily reflect the current date, but rather the most recent date on which the information was last updated. It should also be in a day-month-year format as shown:

AS OF DATE
/15*MAY*78/

The response to the next prompt:

ENTER THE NUMBER OF TASKS PLUS EVENTS
/-----/

should be the total number of types of tasks and events, not the total number of times tasks or events occur.* For example, if there were two tasks, each occurring once during a program, and two events, one occurring twice and the other three times, the total of tasks and events would be four rather than seven. For this example, assume the answer is:

ENTER THE NUMBER OF TASKS PLUS EVENTS
/6----/

SCHEDULE will next inquire:

ENTER THE NUMBER OF REMARKS
/-----/

Any number of remarks may be made (even zero), however, only up to six lines of remarks may be displayed on each page of the report. For this example, the number of remarks will be:

ENTER THE NUMBER OF REMARKS
/1-----/

SCHEDULE is now prepared to accept the input of the various development steps. The screen will blank and display the prompt:

ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS
TASK OR EVENT
FOLLOWED BY NUMBER

*A minimum of four tasks and events is required!

There is no restriction as to the sequence of entering; tasks and events maybe interspersed with one another with different numbers of tasks or events dates. It is necessary, though, when responding to the prompt to identify the element type, either task or event, and the number of elements (i.e., the number of task or event dates). A task is entered by responding with a "T" followed by the number of task dates. The maximum allowable number of task dates is three and if no number is entered, one is assumed. For example, after responding with a "T":

ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS
T

the system will prompt the user for the task I.D. number*, task title, start date, stop date, and percent completion:

ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS
T

```
/ ID# /      TASK      / STR.DT. / STP.DT. / COMPLT /  
/*----*/-----/---*---*/---*---/***-----***/
```

An example response is shown below in which the ID number has been left blank because standard titles are not available and the percent completion is zero**:

ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS
T

```
/ ID# /      TASK      / STR.DT. / STP.DT / COMPLT /  
/*----*/CONCEPT TESTING-----/15*JAN*78/15*MAY*78/*** 0 ***/
```

At this point the system would normally recycle by erasing the screen and asking for the next element type and number of elements. This would not occur for a task with multiple elements (i.e., multiple dates or occurrences). In the event of more than one task date, two for instance, the response to the original prompt would be:

ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS
T2

*The task ID number may be used to identify a standard title to be used for the task title. Standard titles are up to the user and are discussed in detail in Section 3.5.

**Percent completion numbers should be entered as decimals (e.g., 10% = .10, 75% = .75, 100% = 1.).

SCHEDULE would prompt as before for the initial task data, but instead of recycling at this point a request for additional information, namely the second start/stop date and percent completion of the second portion of the task, would be displayed:

```
T2
/ ID# / TASK / STR.DT. / STP.DT / COMPLT /
/*-----*/DEVELOPMENT TESTING-/ 1*JUN*78/ 1*SEP*78/*** 0 ***/
/ STR.DT. / STP.DT / COMPLT /
/--*-----/--*-----/****--***/
```

Once this had been completed as shown:

```
T2
/ ID# / TASK / STR.DT. / STP.DT / COMPLT /
/*-----*/DEVELOPMENT TESTING-/ 1*JUN*78/ 1*SEP*78/*** 0 ***/
/ STR.DT. / STP.DT / COMPLT /
/ 1*OCT*78/ 1*DEC*78/*** 0 ***/
```

the system would recycle and prompt for the next type and number of element. The sequence of operations is essentially the same when entering an event but, of course, the necessary responses are not. An event is identified by an "E" and rather than three, a maximum of nine event dates may be entered. Also, only one date is required and no percent completion. For brevity, two complete operations are shown in Figure 4.2, one for a singular event date and one for a multiple event date. After the system has cycled through either of these processes the number of times equal to the number entered in response to the question about the total number of tasks and events, it will automatically move on to the input of the remarks.* The remark prompt will be displayed:

```
ENTER REMARKS 1

/*-----*/
```

*This assumes that a certain number of remarks (i.e., > 0) were requested. Otherwise this section would be skipped completely.

indicating the remark format and the number of the remark, in this case, one. The remark must be entered after spacing twice as shown below:

ENTER REMARKS 1

/*-----*/

THIS IS A DRAFT SCHEDULE

Since it was indicated earlier that there was only one remark in our example, the system would continue on at this point. If more than one remark had been indicated, the system would recycle that number of times through the remark entering process. Once the input of the remarks is complete, the user will be asked:

DO YOU WANT THIS DATA ADDED TO THE MAIN FILE?

This question should be answered yes. If a no response is given, the data entered up to this point will not be written out to the permanent main data file and control will be returned to the next higher level, thus:

DO YOU WANT THIS DATA ADDED TO THE MAIN FILE?

Y

This does not cause the data to be entered into the main file yet, but it does start the process through which this will be accomplished. The first step is completed by the program which assigns a system file code to the new system and displays it to the user:

DO YOU WANT THIS DATA ADDED TO THE MAIN FILE?

Y

FILE CODE FOR THE SYSTEM IS 0020000

Next, the user will be asked to input the application code to be assigned to all the data that has been inputed:

ENTER APLI CODE

The user has the option of selecting any combination of alphabetic characters, with the exception of the letters A and N. The letter combination "SET" is chosen for this example:

```
ENTER APLI CODE
SET
```

At this point, SCHEDULE will write the new system data out to the master storage file and control will return to the main program which will ask:

DO YOU WISH TO CONTINUE?

A yes answer will continue the session and allow the user to go on to a new task. A no response will terminate the session and return the user to the READY mode:

```
DO YOU WISH TO CONTINUE?
N
```

READY

4.2.2 ENTERING A NEW SYSTEM FROM A FILE

From the READY mode, call up SCHEDULE as shown:

```
READY
EXEC SCHEDULE
```

The screen will erase and the user will be asked:

DO YOU NEED FULL PROMPTING?

to which a yes/no reply is required. A yes response will cause acceptable responses to be displayed after multiple choice questions and a no response will suppress this. An example of both cases is shown in Figure 4.1. Full prompting will be used for this example, so the response to this question is yes ("Y"):

DO YOU NEED FULL PROMPTING?

Y

SCHEDULE will then ask for the system type,

ENTER SYSTEM TYPE

N - NEW (THE SYSTEM DOES NOT HAVE A FILE CODE)

O - OLD (THE SYSTEM IS IN THE MAIN FILE)

which must be responded to with an "N" because a new system is being entered:

ENTER SYSTEM TYPE

N - NEW (THE SYSTEM DOES NOT HAVE A FILE CODE)

O - OLD (THE SYSTEM IS IN THE MAIN FILE)

N

SCHEDULE will next ask to be told where the new data is to be entered from:

WILL YOU ENTER THE DATA FROM THE KEYBOARD OR IS IT PRESENTLY ON A FILE?

ENTER

K - KEYBOARD

F - FILE

N - NONE

Since this example covers entering data from a file, the reply should be:

WILL YOU ENTER THE DATA FROM THE KEYBOARD OR IS IT PRESENTLY ON A FILE?

ENTER

K - KEYBOARD

F - FILE

N - NONE

F

Before proceeding, it would be helpful to briefly cover how the data is originally entered into the file and what format it is in.

SCHEDULE employs two separate data files, the main data file and a buffer file. The main data file is used to permanently store system data; no manipulation of the data is performed within this file other than reading and writing of the records to and from the file. The buffer data file is where the system record data is temporarily stored and data manipulation occurs. Prior to writing a report, the data from the systems to be reported on is written from the main data file to the buffer file in a report format. This means that the data is organized in such a way that will be understood by the graphics section of the SCHEDULE program enabling it to properly display the report.

After a new system has been entered from the keyboard, it is stored in the buffer file in a non-report form in such a way that will be understood by SCHEDULE, allowing it to write the data out to the main data file. If it is ever necessary to copy a system SCHEDULE, or if data entered from the keyboard is not entered into the main data file, it is possible to copy or enter this data, whatever the case may be, without being forced to re-enter all the data. The steps to perform either of these tasks are discussed herein.

4.2.2.1 REPORT FORM

In order to copy a system already in the master file, it must be stored in report form in the buffer file. This is accomplished by following the steps outlined in Section 4.4.1 on writing graphics reports, using the system to be copied up to the point at which the program indicates that the file is properly stored in the buffer file in a report format and is ready to be displayed:

FILE 001 IS READY

At this point SCHEDULE will ask if there is another system to be included in the report and if so what is its system code:

ENTER FILE FOR SYSTEM
XXX0000 - FILE CODE
N - NONE / NO MORE SYSTEMS

It is only possible to copy one report at a time, so the response to this question must be "N":

ENTER FILE FOR SYSTEM
XXX0000 - FILE CODE
N - NONE / NO MORE SYSTEMS
N

SCHEDULE will next inquire where the report is to be printed:

ENTER TERMINAL TYPE
ENTER
G - TEXTRONIX 4014 (GRAPHICAL OUTPUT)
P - TEXAS INSTRUMENTS (PRINTER OUTPUT)
N - NONE

The task of storing the system to be copied in the buffer file in the proper format is accomplished at this point, so there is no reason to display the report (although it may be if desired with no harm being done to the data stored in the buffer file). Thus, the response to the terminal type prompt will be:

ENTER TERMINAL TYPE
ENTER
G - TEXTRONIX 4014 (GRAPHICAL OUTPUT)
P - TEXAS INSTRUMENTS (PRINTER OUTPUT)
N - NONE
N

After completing the above steps, and the steps in the previous section, to arrive at the point at which SCHEDULE is informed that the data is in a file:

WILL YOU ENTER THE DATA FROM THE KEYBOARD OR IS IT PRESENTLY ON A FILE?
ENTER
K - KEYBOARD
F - FILE
N - NONE

F

a prompt will be displayed asking for the format of the data in the buffer file:

IS YOUR DATA IN A REPORT FORM?

Since this was just accomplished, the response is:

IS YOUR DATA IN A REPORT FORM?

Y

SCHEDULE will next want to know:

DO YOU WANT THIS DATA ADDED TO THE MAIN FILE?

If, for some reason, it was not necessary to enter the data into the main file, a negative response would cause the data in the buffer to remain there and control would return to the main program:

DO YOU WANT THIS DATA ADDED TO THE MAIN FILE?

N

DO YOU WISH TO CONTINUE?

from which the session could be terminated or a different task undertaken. After having gone through all these steps, however, it would normally be desirable to respond positively to this question:

DO YOU WANT THIS DATA ADDED TO THE MAIN FILE?

Y

This does not cause the data to be entered into file yet, but starts the process through which this will be accomplished. The first step is completed by the program which assigns a system file code to the new system which is displayed to the user:

DO YOU WANT THIS DATA ADDED TO THE MAIN FILE?

Y

FILE CODE FOR THE SYSTEM IS 0020000

The system will then prompt for up to an eleven character application code:

ENTER APLI CODE

which for the example will be:

ENTER APLI CODE
E

It is at this point, the data is entered into the main data file. When that operation is complete, the system will inquire:

DO YOU WISH TO CONTINUE?

at which time the operator may go on to other tasks by responding "Y" or obtain the READY mode, thusly:

DO YOU WISH TO CONTINUE?
N

READY

4.2.2.2 NON-REPORT FORM

For one reason or another, data which had been previously entered into the buffer file may still be written into the master data file even though it may be after writing reports or even a different schedule TSO session altogether.* Returning to the point left earlier:

WILL YOU ENTER THE DATA FROM THE KEYBOARD OR IS IT PRESENTLY ON A FILE?

ENTER
K - KEYBOARD
F - FILE
N - NONE

*If this is the case, it would be wise prior to executing SCHEDULE to list the buffer data file DATA.DATA to determine that the data in question is still there.

The system will next inquire as before:

IS YOUR DATA IN A REPORT FORM?

Since this is not the case here, the response is:

IS YOUR DATA IN A REPORT FORM?

N

Again, the system will inquire:

DO YOU WANT THIS DATA ADDED TO THE MAIN FILE?

to which a negative answer would cause the data to be left in the buffer file and control to return to the main program. For this example though, the response is:

DO YOU WANT THIS DATA ADDED TO THE MAIN FILE?

Y

It should be pointed out that this step does not cause the data to be entered into the main data file, but rather it initiates the process through which this will be accomplished. The first step is completed by the program which assigns a system file code to the new system which is then displayed to the user:

DO YOU WANT THIS DATA ADDED TO THE MAIN FILE?

Y

FILE CODE FOR THE SYSTEM IS 0030000

Then the system will prompt for an application code to be assigned to all the records:

ENTER APLI CODE

which for the example is:

```
ENTER APLI CODE  
E
```

It is at this point that the data is copied into the main data file for permanent storage. When that operation is complete the system will ask:

```
DO YOU WISH TO CONTINUE?
```

at which time the user may go on to other tasks by responding "Y" or obtain the READY mode by responding "N" as shown:

```
DO YOU WISH TO CONTINUE?  
N
```

```
READY
```

4.3 EDITING OLD SYSTEMS

4.3.1 General

There are presently five edit commands available under the edit mode. The steps leading up to the point where the system inquires as to what type of editing is desired are common to all the commands and will be covered once here at the beginning, rather than repeated for each command. This will be followed by individual discussions of the steps required under each of the commands.

From the READY mode call up the SCHEDULE program as shown:

```
READY  
EXEC SCHEDULE
```

The screen will be erased and SCHEDULE will inquire:

```
DO YOU NEED FULL PROMPTING?
```

to which a yes/no reply is required. A yes response will cause acceptable responses to be displayed after multiple choice questions and a no response will suppress this. An example of both cases is shown in Figure 4.1. Full prompting will be used through all sections of this example, therefore, the response to this question is yes (Y):

DO YOU NEED FULL PROMPTING?
Y

SCHEDULE will next inquire as to the type of system:

ENTER SYSTEM TYPE
N - NEW (THE SYSTEM DOES NOT HAVE A FILE CODE)
O - OLD (THE SYSTEM IS IN THE MAIN FILE)

Since editing may only be performed on data sets presently part of the master storage file, the response to this query must always be old ("O").

ENTER SYSTEM TYPE
N - NEW (THE SYSTEM DOES NOT HAVE A FILE CODE)
O - OLD (THE SYSTEM IS IN THE MAIN FILE)
O

SCHEDULE will then ask:

DO YOU WISH TO WRITE REPORTS OR TO EDIT DATA?
ENTER
E - EDIT DATA
R - WRITE REPORTS
N - NONE

Since it is desired to edit data, the appropriate response is "E":

DO YOU WISH TO WRITE REPORTS OR TO EDIT DATA?
ENTER
E - EDIT DATA
R - WRITE REPORTS
N - NONE
E

This will direct SCHEDULE to the edit mode from which all editing can be accomplished, regardless of what system the record to be edited is in. It will then request that the type of editing required be defined:

ENTER EDIT CODE

- A - ADD
- D - DELETE
- C - CHANGE
- S - SLIP
- R - COPY
- L - LIST
- F - FIND
- N - NONE

The seven editing options listed here are discussed individually in the following sections. The sample report shown in Figure 4.3 will be "edited" to help illustrate practical applications of some of the edit commands, so it might be wise to look at it along with its listing* (Figure 4.4) before continuing, as well as while covering the material contained herein.

4.3.2 Add

After responding with an "A" to the edit code query to indicate that a new record is to be added to a system, the program will ask for the new record number to be assigned to this piece of information:

ENTER RECORD CODE

The response must be the full seven digit code. Suppose for this example it was necessary to insert an event between lines 3 and 4 of the report (Figure 4.3). Referring to the listing, (Figure 4.4) it is seen that the new record number must be somewhere between records number 0010030 and 0010040. Selecting 0010035 for the new record code*:

ENTER RECORD CODE
0010035

*It is not necessary to choose a new code exactly between the old codes, 0010031 could have been used just as well as for the new code.

the system will then inquire as to the type of information that is to be contained in this new record:

```
ENTER ADDITION TYPE
D - SYSTEM DEVELOPMENT STEP
R - SYSTEM REMARK
N - NONE
```

Because system development step refers to either a task or an event, the response is "D", since our example calls for entering an event:

```
ENTER ADDITION TYPE
D - SYSTEM DEVELOPMENT STEP
R - SYSTEM REMARK
N - NONE
D
```

The screen will erase and display the prompt.

```
ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS
TASK OR EVENT
FOLLOWED BY NUMBER
```

As can be seen, it is necessary to identify the element type, either task or event, and the number of elements (i.e., the number of task or event dates). An event is entered by responding with an "E" followed by the number of event dates. The maximum allowable number of event dates is nine and if no number is entered, a one is assumed. Assuming there is only one date associated with the example event the response would be:

```
ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS
TASK OR EVENT
FOLLOWED BY NUMBER
E
```

SCHEDULE then prompts for the event title and event date:

```
ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS
TASK OR EVENT
FOLLOWED BY NUMBER
E
/          TITLE          / DATE /
/-----/-----/---*---*/
```

An example response is shown below:

```
ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS
TASK OR EVENT
FOLLOWED BY NUMBER
E
/          TITLE          / DATE /
/ADD EXAMPLE-----/01*JAN*80/
```

At this point had additional event dates been specified, the system would prompt the user for them; this is not the case in our example. The operation is similar for entering a task except that a task is identified by a "T" and instead of a single date a start and stop date is required along with a percent completion. Also rather than a maximum of 9 dates only up to three start/stop date combinations are permitted. Two complete example operations of adding single and multiple tasks are shown in Figure 4.5 along with the changes to the system record file. Regardless of the type of element added the next prompt will ask the user for the application code to be assigned to this record:

```
/          TITLE          / DATE /
/ADD EXAMPLE-----/01*JAN*80/
APPLICATION CODE
/-----/
```

The user has the option of selecting any combination of alphabetic or numeric characters with the exception of the letters A and N. The letter combination "SET" is chosen for this example:

```
/          TITLE          / DATE /
/ADD EXAMPLE-----/01*JAN*80/
APPLICATION CODE
/SET-----/
```

SCHEDULE will now go ahead and add the new record to the proper system as identified by the system record code. It will indicate when the task is complete and that it has recycled and is ready for another addition:

ADDITION COMPLETE. READY FOR NEXT ADDITION.
ENTER RECORD CODE

If, by mistake, the requested new record code was identical to an existing record code, the program will notify the user and ask for another new code. When a good code is re-entered, the program will add the new record and recycle as above:

ENTER RECORD CODE
0010040
DUPLICATE
RECORD0010040
ENTER RECORD CODE
0010035
ADDITION COMPLETE. READY FOR NEXT ADDITION.
ENTER RECORD CODE

If any more addition were required, they could be made at this time by repeating the process discussed herein. Having completed the additions the add loop can be left by responding with "N" to the record code query:

ADDITION COMPLETE. READY FOR NEXT ADDITION.
ENTER RECORD CODE
N

This will cause the edit code menu to be displayed:

ADDITION COMPLETE. READY FOR NEXT ADDITION.
ENTER RECORD CODE
N
ENTER EDIT CODE
A - ADD
D - DELETE
C - CHANGE
S - SLIP
L - LIST
N - NONE

allowing the user to leave the edit mode (enter "N") or to select another edit mode. As a point of interest, Figure 4.6 is the new system listing which contains the change entered as the example. (See record 0010035.)

For the sake of completeness and assuming there are no further addition, the READY mode can be returned to by entering "N" for the edit code and following these steps:

DO YOU WISH TO WRITE REPORTS OR TO EDIT DATA?

ENTER

E - EDIT DATA

R - WRITE REPORTS

N - NONE

N

DO YOU WISH TO CONTINUE?

N

READY

4.3.3 Delete

After responding with "D" to the edit code query to indicate that a record is to be deleted, the system will prompt the user for the record code of the record to be deleted:

ENTER RECORD CODE

After responding with the seven-digit record code number:

ENTER RECORD CODE

0010035

the program will next indicate that the requested record has been found and successfully deleted and prompt the user for the record code of the next record to be deleted.

DELETION COMPLETE. READY FOR NEXT DELETION.

ENTER RECORD CODE

At this point, the user may continue making deletions or return to the edit menu by entering "N":

```
DELETION COMPLETE.  READY FOR NEXT DELETION.  
ENTER RECORD CODE  
N  
ENTER EDIT CODE  
  A - ADD  
  D - DELETE  
  C - CHANGE  
  S - SLIP  
  L - LIST  
  N - NONE
```

If, by mistake, the user entered a record code that was not in the master file, that system would indicate that it was unable to find the requested record and prompt the user for another record code*:

```
ENTER RECORD CODE  
0010033  
0010033RECORD NOT FOUND  
ENTER RECORD CODE
```

When the edit menu is obtained, the user may request another edit mode or obtain the READY mode by typing "N" and completing the following steps:

```
DO YOU WISH TO WRITE REPORTS OR TO EDIT DATA?  
ENTER  
  E - EDIT DATA  
  R - WRITE REPORTS  
  N - NONE  
N  
DO YOU WISH TO CONTINUE?  
N  
READY
```

*It would be best at this point to return to the edit menu and obtain a listing of the system which contains the record in question.

4.3.4 List

After responding with an "L" to the edit code query to indicate that a listing is desired, the program will offer the option of three different types of listings:

```
ENTER LIST TYPE
A- SYSTEM HEADING RECORDS
S - ALL RECORDS OF SYSTEM X
P - SUBSET OF ALL RECORDS OF SYSTEM X
N - NONE
```

Each of these options is discussed separately herein.

System Heading Records

Responding to the list type query with an "A":

```
ENTER LIST TYPE
A- SYSTEM HEADING RECORDS
S - ALL RECORDS OF SYSTEM X
P - SUBSET OF ALL RECORDS OF SYSTEM X
N - NONE
```

A

will provide a listing giving the System Number, Major System, Project Title, and As-Of-Date as shown below for all the systems contained in the master data file.

```
SYSTEM NUMBER: 001
MAJOR SYSTEM:  DEV COMAND BUSINESS SYSTEMS
PROJECT TITLE:  SCHEDULES
AS OF DATE:   12 AUG 77
```

After all the information on all the systems has been listed, the program will return with the next unused system record code and the list type query:

```
00200000RECORD NOT FOUND
ENTER LIST TYPE
A - SYSTEM HEADING RECORDS
S - ALL RECORDS OF SYSTEM X
P - SUBSET OF ALL RECORDS OF SYSTEM X
N - NONE
```

All Records of System X.

Responding to the list type query with an "S":

ENTER LIST TYPE

- A - SYSTEM HEADING RECORDS
- S - ALL RECORDS OF SYSTEM X
- P - SUBSET OF ALL RECORDS OF SYSTEM X
- N - NONE

S

will provide a listing of all the records contained within the system identified in the next prompt question:

ENTER SYSTEM CODE

It is only necessary to enter the system number (first three digits of the system code) in response to this question:

ENTER SYSTEM CODE

001

SCHEDULE will then display a listing which gives the heading date, all development steps and all the remarks of the requested system. This is the command which was used to obtain the listing shown in Figure 4.4. This is immediately followed by the list type query:

ENTER LIST TYPE

- A - SYSTEM HEADING RECORDS
- S - ALL RECORDS OF SYSTEM X
- P - SUBSET OF ALL RECORDS OF SYSTEM X
- N - NONE

Subset of All Records of System X

Responding to this query with a "P":

ENTER LIST TYPE

A - SYSTEM HEADING RECORDS

S - ALL RECORDS OF SYSTEM X

P - SUBSET OF ALL RECORDS OF SYSTEM X

N - NONE

P

will allow the user to look at a specific subset of records within a system identified by a particular application code. After the above response is made, the program will ask the user what system is to be examined:

ENTER SYSTEM CODE

The response should be a three digit number identifying the system number:

ENTER SYSTEM CODE

001

Next, the program will inquire for the aplicode code (aplicode) that will identify the records within the system that are to be displayed. Remember that the aplicode entered must match the code within the record in both letters and sequence in order to be displayed:

ENTER APPLICATION CODE

B - BUDGET

E - PROJECT ENGINEER

H - PROJECT MANAGER

O - 104 REPORT

P - PROCUREMENT

Q - PRODUCT ASSURANCE

R - RECAP

S - PROGRAM SUMMARY

T - TEST

Referring to the example listing in Figure 4.7, it is seen that all the records have an aplice of "E" and within that set, two of the records have an aplice of "SET." If these last two records were of interest, they could be listed by responding to the prompt with:

```
ENTER APPLICATION CODE
B - BUDGET
E - PROJECT ENGINEER
H - PROJECT MANAGER
O - 104 REPORT
P - PROCUREMENT
Q - PRODUCT ASSURANCE
R - RECAP
S - PROGRAM SUMMARY
T - TEST
SET
```

This would produce a listing as shown in Figure 4.8.:

If instead, for example, it was necessary to list all records with an aplice of "E", the response to the aplice query would be:

```
ENTER APPLICATION CODE
B - BUDGET
E - PROJECT ENGINEER
H - PROJECT MANAGER
O - 104 REPORT
P - PROCUREMENT
Q - PRODUCT ASSURANCE
R - RECAP
S - PROGRAM SUMMARY
T - TEST
E
```

This would produce the listing contained in Figure 4.9. Notice that the records whose application code is SET are also listed because the aplice code requested "T" is a subset of "SET". The end of the listing is indicated by the phrase "END OF MAIN FILE", which is immediately followed by the list type query:

```
ENTER LIST TYPE
A - SYSTEM HEADING RECORDS
S - ALL RECORDS OF SYSTEM X
P - SUBSET OF ALL RECORDS OF SYSTEM X
N - NONE
```

When all the lists have been compiled, the user can respond with an "N" to the list type query which will cause the edit code menu to be displayed:

ENTER EDIT CODE

- A - ADD
- D - DELETE
- C - CHANGE
- S - SLIP
- L - LIST
- N - NONE

Allowing the user to leave the edit mode by entering "N" or to select another edit mode. If the editing was complete, the user could return to the READY mode by completing these steps:

ENTER EDIT CODE

- A - ADD
- D - DELETE
- C - CHANGE
- S - SLIP
- L - LIST
- N - NONE

N

DO YOU WISH TO WRITE REPORTS OR TO EDIT DATA?

ENTER

- E - EDIT DATA
- R - WRITE REPORTS
- N - NONE

N

DO YOU WISH TO CONTINUE?

N

READY

4.3.5 Change

After responding with a "C" to the edit code query to indicate that a change is to be made SCHEDULE will prompt the user for the record code number of the record to be changed:

ENTER RECORD CODE

The response must be the full seven digit record code number. There is no need to differentiate between a development step record, a remark record, or a heading record, SCHEDULE is able to do this on its own and choose the correct logic path. Changes to all of these record types are discussed in the following sections.

4.3.5.1 DEVELOPMENT STEP RECORD CHANGES

Referring to the example system listing in Figure 4.4 (System number 001) and assuming that it is necessary to make changes to record number 80, the response to the record code query is:

```
ENTER RECORD CODE
0010080
```

SCHEDULE will recognize this record as an event and prompt the user to identify the portion of the record to be changed:

```
ENTER INFORMATION TYPE TO BE CHANGED
ENTER
  T - TITLE
  D - DATA
  A - APPLICATION CODE
  E - ELEMENT TYPE
  N - NONE
```

If instead the record had been a task, SCHEDULE would display the same prompt, however, some of the options would be slightly different in their purpose. This will be covered in the discussion of the individual change types contained herein.

4.3.5.1.1 TITLE

After responding with a "T" to indicate that a change is to be made to the title, the program will list the development step type, in this case event, the current title, and the allowable format in which the new title can be entered:

```
T
/          EVENT          /
THIS IS A 31 DEC 76 EVENT
/-----/
```

After entering the new title:

```
/          EVENT          /  
THIS IS A 31 DEC 76 EVENT  
/THIS IS A NEW TITLE-----/
```

The system will recycle and redisplay the change menu. At this point, the user may type "N" in which case the program will indicate that the new change has been entered and then prompt for a new record code:

ENTER INFORMATION TYPE TO BE CHANGED

ENTER

```
T - TITLE  
D - DATA  
A - APPLICATION CODE  
E - ELEMENT TYPE  
N - NONE
```

N

CHANGE COMPLETE. READY FOR NEXT CHANGE.

ENTER RECORD CODE

or select a new type of change.

4.3.5.1.2 DATA

After responding with a "D" and depending on the type of development step that the record contains, the start date/stop date/percent completion or event data can be changed (The former refers to a task and the latter to an event). The basic process is the same for either case with the information type displayed along the current date and correct input format, thus, for this example of an event record, the prompt would be:

```
D  
/ EVENT DATE /  
31*DEC*76  
/--*---*--/
```


and in the case of a task record (See Figure 4.4, RECORD 0090) it would be:

```
D
/ STR.DT. / STP.DT. / COMPLT /
31*DEC*75 09*JAN*78 0.60
/-----*/-----*/-----*/
```

After entering a new date into the example record:

```
/ EVENT DATE /
31*DEC*76
/ 1*JAN*77/
```

the system will recycle and redisplay the change menu. At this point, the user may type "N" in which case the program will indicate that new change has been entered and then prompt for a new record code number:

```
ENTER INFORMATION TYPE TO BE CHANGED
ENTER
T - TITLE
D - DATA
A - APPLICATION CODE
E - ELEMENT TYPE
N - NONE
N
CHANGE COMPLETE. READY FOR NEXT CHANGE
ENTER RECORD CODE
```

or select a new change type.

4.3.5.1.3 APPLICATION CODE

After responding with an "A" to indicate that a change is to be made to the application code of the record, the program will display the current application code contained in the record and the allowable format for the new application code.

```
A
E
/-----/
```

This will be the same for either development step type. After responding with the new application code:

E
/T-----/

the system will recycle and redisplay the change menu. At this point, the user may type "N", in which case the program will indicate that the new change has been entered and then prompt for a new record code:

ENTER INFORMATION TYPE TO BE CHANGED

ENTER

T - TITLE

D - DATA

A - APPLICATION CODE

E - ELEMENT TYPE

N - NONE

N

CHANGE COMPLETE. READY FOR NEXT CHANGE

ENTER RECORD CODE

or select a new change type.

4.3.5.1.4 ELEMENT CODE

This is a very useful change step in that it allows the user to change an event to a task and visa-versa. After responding with an "E", the program will display the current element and prompt the user for the new element type and number of elements.

E

THE ELEMENT TITLE IS ... THIS IS A NEW TITLE

ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS

TASK OR EVENT

FOLLOWED BY NUMBER

A change to a task is accomplished by entering "T", followed by the number of individual elements, up to a maximum of three, within the task, if it is greater than one. The system will then prompt for the start date, stop date, and percent complete:

```
T
/ STR.DT. / STP.DT. / COMPLT /
/--*---*--/--*---*--/***---***/
```

After entering the requested data, the program will either recycle and request the same data a number of times equal to the number of elements previously entered, or display the change type menu.

A change to an event is accomplished by entering "E", followed by the number of individual event dates, up to a maximum of nine if greater than one. The system will then prompt for the event date:

```
E
/ EVENT DATE /
/--*---*--/
```

After entering the new date, the program will either recycle and request the same information equal to the number of dates entered above or display the change menu.

From the change menu, the user may type "N", in which case, the program will indicate that the change has been entered and then prompt for a new record code:

```
ENTER INFORMATION TYPE TO BE CHANGED
ENTER
  T - TITLE
  D - DATA
  A - APPLICATION CODE
  E - ELEMENT TYPE
  N - NONE
N
CHANGE COMPLETE.  READY FOR NEXT CHANGE
ENTER RECORD CODE
```

or select a new change.

4.3.5.2 REMARKS RECORD CHANGES

Referring to the example system listing in Figure 4.4 (System Number 001) and assuming it is necessary to make changes to Record 170, the response to the record code query is:

```
ENTER RECORD CODE
0010170
```

The system will recognize this record as a remark and prompt the user to identify the portion of the record to be changed.

```
ENTER ITEM TO BE CHANGED
R - REMARK
A - APPLICATION CODE
N - NONE
```

A detailed discussion of these options follows.

4.3.5.2.1 REMARK

After responding with an "R" to indicate the remark itself is to be changed, the system will prompt the user for the new remark by displaying the remark currently contained in the record and the format to be followed when entering the new remark:

```
R
ENTER REMARKS
MIDWEST SCIENTIFIC & ENGINEERING COMPUTER.
/*-----*/
```

After spacing over twice and entering the new record as shown:

```
MIDWEST SCIENTIFIC & ENGINEERING COMPUTER.
/*-----*/
SOUTHWEST SCIENTIFIC & ENGINEERING COMPUTER
```


The system will recycle and display the change menu. At this point, the user may type "N", in which case the program will indicate that the new change has been entered and then prompt for a new record code:

ENTER ITEM TO BE CHANGED

R - REMARK

A - APPLICATION CODE

N - NONE

N

CHANGE COMPLETE. READY FOR NEXT CHANGE.

ENTER RECORD CODE

or elect to also change the application code.

4.3.5.2.2 APPLICATION CODE

After responding with an "A" to indicate that the application code of the remark is to be changed, the program will display the current application code of the record and the proper format in which the new code should be entered:

A

E

/-----/

After entering the new application code:

A

E

/T-----/

the system will recycle and display the change menu. At this point, the user may type "N", in which case the program will indicate that the change has been entered and then prompt for a new record code:

ENTER ITEM TO BE CHANGED

R - REMARK

A - APPLICATION CODE

N - NONE

N

CHANGE COMPLETE. READY FOR NEXT CHANGE.

ENTER RECORD CODE

or elect to also change the remark.

4.3.5.3 HEADER INFORMATION CHANGES

Referring to the example system listing in Figure 4.3* (System Number 001) and assuming it is necessary to make changes to the heading information contained in Record 0000, the response to the record code query is:

```
ENTER RECORD CODE
0010000
```

The system will recognize this as a header card and prompt the user to identify what item in the header data is to be changed.

```
ENTER HEADING ITEM TO BE CHANGED
M - AIRCRAFT SYSTEM
T - SYSTEM TITLE
D - AS OF DATE
E - START YEAR FOR REPORT
R - NUMBER OF YEARS IN REPORT
A - APPLICATION CODE
N - NONE
```

Detailed discussions of these options follow.

4.3.5.3.1 AIRCRAFT SYSTEM

After responding with an "M" to indicate that a change is to be made to the aircraft system title, the system will prompt the user by displaying the current title and the correct format for the new title:

```
M
ENTER MAJOR SYSTEMS
DEV COMMAND BUSINESS SYSTEMS
/-----/-----/
```

*All the information contained in the header record is not displayed in this type listing. A systems header listing is a better source for this information.

After entering the new title:

```
M
ENTER MAJOR SYSTEMS
DEV COMMAND  BUSINESS SYSTEMS
/NEW COMMAND /NEW BUSINESS-----/
```

the system will recycle and display the change menu. At this point, the user can type "N", whereupon the program will indicate that the change has been entered and prompt the user for a new record code:

```
ENTER HEADING ITEM TO BE CHANGED
M - AIRCRAFT SYSTEM
T - SYSTEM TITLE
D - AS OF DATE
E - START YEAR FOR REPORT
R - NUMBER OF YEARS IN REPORT
A - APPLICATION CODE
N - NONE
NN
CHANGE COMPLETE.  READY FOR NEXT CHANGE.
ENTER RECORD CODE
```

or select a new type of change.

4.3.5.3.2 SYSTEM TITLE

After responding with a "T" to indicate that a change must be made to the system title, the system will prompt the user by displaying the current title and the correct format for the new system title:

```
T
ENTER TITLE
SCHEDULES
/-----/-----/
```

After entering the new system title:

```
ENTER TITLE
SCHEDULES
/NEW SCHEDULES-----/-----/
```

the program will recycle and display the change menu. At this time, the user can type "N", in which case the system will indicate that the change has been made and prompt the user for a new record code:

ENTER HEADING ITEM TO BE CHANGED

M - AIRCRAFT SYSTEM

T - SYSTEM TITLE

D - AS OF DATE

E - START YEAR FOR REPORT

R - NUMBER OF YEARS IN REPORT

A - APPLICATION CODE

N - NONE

N

CHANGE COMPLETE. READY FOR NEXT CHANGE.

ENTER RECORD CODE

or select a new type of change.

4.3.5.3.3 AS-OF-DATE

After responding with "D" to indicate that a change is to be made to the As-of-Date, the program will prompt the user by displaying the current As-of-Date and the correct format for the new As-of-Date:

D

AS OF DATE

12 AUG 77

/--*---*---/

After the new date has been entered:

D

AS OF DATE

12 AUG 77

/ 1*JUN*78/

the program will recycle and display the change menu. At this point, the user can either type "N", in which case, the system will indicate when the change has been entered and prompt the user for a new record code:

ENTER HEADING ITEM TO BE CHANGED

M - AIRCRAFT SYSTEM

T - SYSTEM TITLE

D - AS OF DATE

E - START YEAR FOR REPORT

R - NUMBER OF YEARS IN REPORT

A - APPLICATION CODE

N - NONE

N

CHANGE COMPLETE. READY FOR NEXT CHANGE.

ENTER RECORD CODE

or select a new change option.

4.3.5.3.4 START YEAR FOR REPORT

After responding with "E" to indicate the starting year for the report is to be changed, the system will prompt the user by displaying the current start year in the record and the format for the new starting year:

E

ENTER START YEAR FOR REPORT

11

/-----/

After the last two figures of the new starting year have been entered:

E

ENTER START YEAR FOR REPORT

11

/78---/

or select a new change option.

4.3.5.3.5 NUMBER OF YEARS IN REPORT

After responding with "R" to indicate there is a need to change the number of years in report value in the heading information, the system will prompt the user by displaying the current number of years in the report and the correct format for the new value:

```
R
ENTER NUMBER OF YEARS IN REPORT
  6
/-----/
```

Remember that this number may not be larger than 15.

After entering the new value:

```
R
ENTER NUMBER OF YEARS IN REPORT
  6
/8-----/
```

the system will recycle and display the change menu. At this point, the user can either type "N", whereupon the system will indicate when the change has been entered and prompt the user for a new record code:

```
ENTER HEADING ITEM TO BE CHANGED
M - AIRCRAFT SYSTEM
T - SYSTEM TITLE
D - AS OF DATE
E - START YEAR FOR REPORT
R - NUMBER OF YEARS IN REPORT
A - APPLICATION CODE
N - NONE
```

```
N
CHANGE COMPLETE.  READY FOR NEXT CHANGE.
ENTER RECORD CODE
```

or select a new change option.

4.3.5.3.6 APPLICATION CODE

After entering "A" to indicate there is to be a change to the application code of the header card data, the system will prompt the user by displaying the current application code and the correct format for the new code:

```
A
      E
/-----/
```

After entering a new application code:

```
A
      E
/T-----/
```

the program will recycle and display the change menu. At this time, the user may either type "N", whereupon the system will indicate when the change has been entered and prompt the user for a new record code:

ENTER HEADING ITEM TO BE CHANGED

M - AIRCRAFT SYSTEM

T - SYSTEM TITLE

D - AS OF DATE

E - START YEAR FOR REPORT

R - NUMBER OF YEARS IN REPORT

A - APPLICATION CODE

N - NONE

N

CHANGE COMPLETE. READY FOR NEXT CHANGE.

ENTER RECORD CODE

or select a new change option.

4.3.5.4 RETURN

After all changes have been made, rather than responding to the record query with the record number of the next record to be changed, enter "N". This will return control to the edit menu.

ENTER RECORD CODE

N

ENTER EDIT DATA

A - ADD

D - DELETE

C - CHANGE

S - SLIP

L - LIST

N - NONE

From this point, the user may request another edit mode or move on to another task. The READY mode may be obtained by completing these steps:

ENTER RECORD CODE

N

ENTER EDIT CODE

A - AD

D - DELETE

C - CHANGE

S - SLIP

L - LIST

N - NONE

N

DO YOU WISH TO WRITE REPORTS OR TO EDIT DATA?

ENTER

E - EDIT DATA

R - WRITE REPORTS

N - NONE

N

DO YOU WISH TO CONTINUE?

N

READY

4.3.6 Slip

After responding with an "S" to the edit code query to indicate that some or even all of the SCHEDULE data for a system is to be slipped (i.e., moved back in time) the program will ask for the system code of the system SCHEDULE on which this action is to be performed:

S
ENTER SYSTEM CODE

The response need only be the three digit system code, which is the first three digits of the record code for each record in the system. Using the example system code 001:

ENTER SYSTEM CODE
001

The system will next prompt the user to determine how many records are to be slipped:

HOW MANY RECORDS DO YOU WANT TO SLIP?
A - ALL DATES (START&STOP) THAT OCCUR AFTER DATE X
S - ONLY STOP DATES THAT OCCUR AFTER DATA X
N - NONE

A response of "A" will cause all dates, that is, start, stop, and event dates, after date x (which will be entered in a moment) to be slipped. "S" will cause only the stop and event dates after date x, to be slipped. Assuming the example system in Figure 4.10 has a requirement to slip all the dates the response is:

HOW MANY RECORDS DO YOU WANT TO SLIP?
A - ALL DATES (START&STOP) THAT OCCUR AFTER DATE X
S - ONLY STOP DATES THAT OCCUR AFTER DATA X
N - NONE

A

The program will then prompt for the reference date x after which all the slips will take place:

ENTER SLIP REFERENCE DATE X
/--*---*--/

After entering the reference date x:

ENTER SLIP REFERENCE DATE X
/ 1*JAN*82/

The system will inquire:

SHOULD THE SLIP DEPEND ON APLI CODE?

ENTER

Y - YES

N - NO

This gives the user a certain degree of selectivity over which record are to be slipped rather than being forced to change all the records of a system. If a yes response is given, the user will be asked for the application code of the records to be slipped:

Y

ENTER APLICODE FOR STEPS TO BE SLIPPED
/-----/

A no response will simply bypass this question. In either case, SCHEDULE will next ask for:

ENTER NUMBER OF MONTHS OF SLIPPAGE
/----/

The reponse to this question must be the slippage in months in a decimal format. For example, a two year slip is 24. months, while a two week slip is .5 months. If the example system slip was six months, the response would be:

ENTER NUMBER OF MONTHS OF SLIPPAGE
/6.--/

To avoid any mistakes (Slip cannot be reversed short of manually changing all the dates inadvertently moved back) SCHEDULE will display the number it has read in decimal format and ask if it is correct:

ENTER NUMBER OF MONTHS OF SLIPPAGE

/6.--/

6.00

DO WE AGREE ON THE NUMBER OF MONTHS?

If the original response had not been six months the no response ("N") will cause the system to recycle and allow the user to re-enter the number of months to be slipped:

DO WE AGREE ON THE NUMBER OF MONTHS?

N

ENTER NUMBER OF MONTHS OF SLIPPAGE

/----/

six is correct, so a positive response (i.e., yes) is appropriate.

DO WE AGREE ON THE NUMBER OF MONTHS?

Y

Once this yes response is received, SCHEDULE will go ahead and make the slips in the system SCHEDULE as instructed. Figure 4.10 is the original SCHEDULE and Figure 4.11 is the slipped SCHEDULE. When completed, the edit menu is displayed:

DO WE AGREE ON THE NUMBER OF MONTHS?

Y

ENTER EDIT CODE

A - ADD

D - DELETE

C - CHANGE

S - SLIP

L - LIST

N - NONE

allowing the user to leave the edit mode (enter "N") or to select a new edit mode. To obtain the "READY" mode from this point, complete the following steps:

ENTER EDIT CODE

A - ADD
D - DELETE
C - CHANGE
S - SLIP
L - LIST
N - NONE

N

DO YOU WISH TO WRITE REPORTS OR TO EDIT DATA?

ENTER

E - EDIT DATA
R - WRITE REPORTS
N - NONE

N

DO YOU WISH TO CONTINUE?

N

READY

4.4 WRITE REPORTS

4.4.1 Graphic Report

From the READY mode, call up SCHEDULE as shown:

READY
EXEC SCHEDULE

The screen will clear and the system will inquire:

DO YOU NEED FULL PROMPTING?

to which a yes/no reply is required. A yes response will cause acceptable responses to be displayed after multiple choice questions and a no response will suppress this. An example of both cases is shown in Figure 4.1. Full prompting will be used for this example, so the reply to this question is yes (Y):

DO YOU NEED FULL PROMPTING?
Y

SCHEDULE will then inquire as to the system type:

ENTER SYSTEM TYPE
N - NEW (THE SYSTEM DOES NOT HAVE A SYSTEM CODE)
O - OLD (THE SYSTEM IS IN THE MAIN FILE)

which must be responded to with old since report writing must be performed on old (i.e., previously entered) data sets, thus:

ENTER SYSTEM TYPE
N - NEW (THE SYSTEM DOES NOT HAVE A SYSTEM CODE)
O - OLD (THE SYSTEM IS IN THE MAIN FILE)
O

Having identified the system type, the task to be performed with the data set must be stated:

DO YOU WISH TO WRITE REPORTS OR TO EDIT DATA?
ENTER
E - EDIT DATA
R - WRITE REPORTS
N - NONE

Obviously, to write reports, the response is "R":

DO YOU WISH TO WRITE REPORTS OR TO EDIT DATA?
ENTER
E - EDIT DATA
R - WRITE REPORTS
N - NONE
R

Once the system knows that reports are to be written, it must be told which systems are to be reported on:

HOW MANY SYSTEMS DO YOU WANT TO SEE?
ENTER
A - ALL
N - NONE
S - SUBSET OF ALL

Answering "A" would result in a report for all the systems contained in the accessed dataset. For this example, the response will be "S", which will permit selective choosing of the system(s) to be reported on in a later step, (If "A" was the response in this case there would be no query at a later step to identify the systems to be reported on.) hence;

HOW MANY SYSTEMS DO YOU WANT TO SEE?

ENTER

A - ALL

N - NONE

S - SUBSET OF ALL

S

The system will then prompt the user for a name up to 20 characters long of the office preparing the reports:

ENTER OFFICE PREPARING REPORTS

Per the example report this is entered as:

ENTER OFFICE PREPARING REPORTS

SECO

Recall that earlier it was requested to report on a selected subset of systems, these systems must now be identified individually by file code:

ENTER SYSTEM CODE

XXX0000 - SYSTEM CODE

N - NONE / NO MORE SYSTEMS

For this example, only one file will be requested for viewing as:

ENTER SYSTEM CODE

XXX0000 - SYSTEM CODE

N - NONE / NO MORE SYSTEMS

0010000

It is not required to enter all digits because the program is only looking for the first three digits to which it automatically adds the remaining four zeros. It is possible to narrow down the scope of the report further by specifying that only records with a particular application code (APLICODE) be displayed. (These codes are assigned at the time the system is entered into the master storage file). Care must be taken that the APLICODE requested here match in both letters and sequence the APLICODE of the record as it appears in the data file.

ENTER (OFFICE/SUBJECT/REPORT/APLICODE) OF INTEREST

- A - ALL
- B - BUDGET
- E - PROJECT ENGINEER
- H - PROJECT MANAGER
- O - 104 REPORT
- P - PROCUREMENT
- Q - PRODUCT ASSURANCE
- R - RECAP
- S - PROGRAM SUMMARY
- T - TEST
- N - NONE

The response in this example will be "A", which will cause all the records contained in the system data file to be included in the report rather than a particular subset.

ENTER (OFFICE/SUBJECT/REPORT/APLICODE) OF INTEREST

- A - ALL
- B - BUDGET
- E - PROJECT ENGINEER
- H - PROJECT MANAGER
- O - 104 REPORT
- P - PROCUREMENT
- Q - PRODUCT ASSURANCE
- R - RECAP
- S - PROGRAM SUMMARY
- T - TEST
- N - NONE

A

The screen will again blank and the system will ask that a report title, up to 40 characters long, be entered:

ENTER REPORT TITLE

Figure 4.12 is a copy of an example report that will be recreated here, whose title is entered as shown:

ENTER REPORT TITLE
SAMPLE REPORT (GRAPHICS)

The system will indicate that the requested file is prepared and prompt for another file code to which the reply "N" will be given to indicate that no more systems are to be included in the report:

ENTER SYSTEM CODE
XXX0000 - SYSTEM CODE
N - NONE / NO MORE SYSTEMS
0010000
FILE 001 IS READY.
ENTER SYSTEM CODE
XXX0000 - SYSTEM CODE
N - NONE / NO MORE SYSTEMS
N

If the reply to the inquiry covering the number of systems had been "ALL", instead of asking for the file codes, the system would search on its own indicating what files had been processed for reporting until an end-of-file condition is reached. For this example, this would appear as;

ENTER OFFICE PREPARING REPORTS
SECO
FILE 001 IS READY.
0020000RECORD NOT FOUND

Regardless of the number of files requested in each case, the next prompt is:

ENTER TERMINAL TYPE
ENTER
G - TEXTRONIX 4014 (GRAPHICAL OUTPUT)
P - TEXAS INSTRUMENTS (PRINTER OUTPUT)
N - NONE

The response must indicate the type of terminal on which the report will be displayed, here a Textronix 4014, thus:

ENTER TERMINAL TYPE

ENTER

G - TEXTRONIX 4014 (GRAPHICAL OUTPUT)

P - TEXAS INSTRUMENTS (PRINTER OUTPUT)

N - NONE

G

The system next displays a menu which offers three plotting options and a return:

TYPE C FOR CONTINUOUS PLOTS

TYPE R TO RERUN PLOTS

TYPE S TO RETURN TO MAIN PROGRAM

HIT SPACE BAR FOR INDIVIDUAL PLOTTING

"C" will continuously plot all the reports and automatically make a copy of each page. "R" permits a replotting of all the reports without having to return to the main program and re-entering the required information. "S" returns control to the main program. The space bar (i.e., blank character) allows the user to go through the requested reports and manually make a copy if desired. The response for this example will be the space bar.

SCHEDULE must now prompt for two final pieces of information, first:

ENTER FORM START DATE FOR:EXAMPLE OF A REPORT

?

This request is for the last digits of the calendar year which will appear at the left-hand margin of the time-frame portion of the report. Presently, this number must be larger than 76 and need not reflect the current date or fiscal year. The current date is displayed as a vertical dotted line on the report form automatically (see Figure 4.12). For this example, the form will be started with CY77:

ENTER FORM START DATE FOR:EXAMPLE OF A REPORT

?

77

The next and last prompt prior to the actual plotting of the graph(s) is:

```
ENTER NUMBER OF YEARS IN FORM (1-15)
?
```

The reply must fall within the limits shown, and for this example, will be nine years:

```
ENTER NUMBER OF YEARS IN FORM (1-15)
?
9
```

At this point, the system will begin producing the reports in the requested format. When all the requested systems have been plotted, the plotting format menu will again be displayed, offering the same options. To complete the example, "S" would be entered to return control to the main program and by responding "NONE" to the remaining prompts, the READY mode will be obtaining.

4.4.2 Typewriter Report

From the READY mode, call up SCHEDULE as shown:

```
READY
EXEC SCHEDULE
```

The screen will clear and the system will inquire:

```
DO YOU NEED FULL PROMPTING?
```

to which a yes/no reply is required. A yes response will cause acceptable responses to be displayed after multiple choice questions and a no response will suppress this. An example of both cases is shown in Figure 4.1. Full prompting will be used for this example, so the reply to this question is yes (Y):

```
DO YOU NEED FULL PROMPTING?
Y
```

Answering "A" would result in a report for all the systems contained in the accessed dataset. For this example, the response will be "S", which will permit selective choosing of SCHEDULE will then inquire as to the system type:

ENTER SYSTEM TYPE

- N - NEW (THE SYSTEM DOES NOT HAVE A SYSTEM CODE)
- O - OLD (THE SYSTEM IS IN THE MAIN FILE)

which must be responded to with old since report writing must be performed on old (i.e., previously entered) data sets, thus:

ENTER SYSTEM TYPE

- N - NEW (THE SYSTEM DOES NOT HAVE A SYSTEM CODE)
- O - OLD (THE SYSTEM IS IN THE MAIN FILE)

O

Having identified the system type, the task to be performed with the data set must be stated:

DO YOU WISH TO WRITE REPORTS OR TO EDIT DATA?

ENTER

- E - EDIT DATA
- R - WRITE REPORTS
- N - NONE

Obviously, to write reports, the response is "R":

DO YOU WISH TO WRITE REPORTS OR TO EDIT DATA?

ENTER

- E - EDIT DATA
- R - WRITE REPORTS
- N - NONE

R

Once the system knows that reports are to be written, it must be told which systems are to be reported on:

HOW MANY SYSTEMS DO YOU WANT TO SEE?

ENTER

- A - ALL
- N - NONE
- S - SUBSET OF ALL

Answering "A" would result in a report for all the systems contained in the accessed dataset. For this example, the response will be "S", which will permit selective choosing of the system(s) to be reported on in a later step, hence:

HOW MANY SYSTEMS DO YOU WANT TO SEE?

ENTER

A - ALL

N - NONE

S - SUBSET OF ALL

S

The system will then prompt the user for a name up to 20 characters long of the office preparing the reports:

ENTER OFFICE PREPARING REPORTS

Per the example report this is entered as:

ENTER OFFICE PREPARING REPORTS

SECO

Recall that earlier it was requested to report on a selected subset of systems, these systems must now be identified individually by file code:

ENTER SYSTEM CODE

XXX0000 - SYSTEM CODE

N - NONE / NO MORE SYSTEMS

For this example, only one file will be requested for viewing as:

ENTER SYSTEM CODE

XXX0000 - SYSTEM CODE

N - NONE / NO MORE SYSTEMS

0010000

It is not required to enter all digits because the program is only looking for the first three digits to which it will automatically add the remaining four zeros. It is possible to narrow down the scope of the report further by specifying that only records with a particular application code (APLICODE) be displayed. (These codes are assigned at the time the system is entered into the master storage file.) Care must be taken that the APLICODE requested here match in both letters and sequence the APLICODE of the record as it appears in the data file.

ENTER (OFFICE/SUBJECT/REPORT/APLICODE) OF INTEREST

- A - ALL
- B - BUDGET
- E - PROJECT ENGINEER
- H - PROJECT MANAGER
- O - 104 REPORT
- P - PROCUREMENT
- Q - PRODUCT ASSURANCE
- R - RECAP
- S - PROGRAM SUMMARY
- T - TEST
- N - NONE

The response in this example will be "A", which will cause all the records contained in the system data file to be included in the report rather than a particular subset.

ENTER (OFFICE/SUBJECT/REPORT/APLICODE) OF INTEREST

- A - ALL
- B - BUDGET
- E - PROJECT ENGINEER
- H - PROJECT MANAGER
- O - 104 REPORT
- P - PROCUREMENT
- Q - PRODUCT ASSURANCE
- R - RECAP
- S - PROGRAM SUMMARY
- T - TEST
- N - NONE

A

The screen will again blank and the system will ask that a report title, up to 40 characters long, be entered:

ENTER REPORT TITLE

Figure 4.13 is a copy of an example report that will be recreated here, whose title is entered as shown:

```
ENTER REPORT TITLE
SAMPLE REPORT (PRINTER)
```

The system will indicate that the requested file is prepared and prompt for another file code; to indicate that no more systems are to be included in the report the reply "N" will be given:

```
ENTER SYSTEM CODE
XXX0000 - SYSTEM CODE
N - NONE / NO MORE SYSTEMS
0010000
FILE IS READY.
ENTER SYSTEM CODE
XXX0000 - SYSTEM CODE
N - NONE / NO MORE SYSTEMS
N
```

If the reply to the inquiry covering the number of systems had been "ALL", instead of asking for the file codes, the system would search on its own indicating what files had been processed for reporting until an end-of-file condition is reached. For this example, this would appear as;

```
ENTER OFFICE PREPARING REPORTS
SECO
FILE 001 IS READY.
0020000RECORD NOT FOUND
```

Regardless of the number of files requested in each case, the next prompt is:

```
ENTER TERMINAL TYPE
ENTER
G - TEXTRONIX 4014 (GRAPHICAL OUTPUT)
P - TEXAS INSTRUMENTS (PRINTER OUTPUT)
N - NONE
```

The response must indicate the type of terminal on which the report will be displayed, here a Textronix 4014, thus:

ENTER TERMINAL TYPE

ENTER

G - TEXTRONIX 4014 (GRAPHICAL OUTPUT)
P - TEXAS INSTRUMENTS (PRINTER OUTPUT)
N - NONE

P

At this point SCHEDULE will promptly begin to print out the report in the format shown in Figure 4.13.

4.5 SPECIAL OPTIONS

4.5.1 General

This section discusses options which are beyond the scope of the purpose of the first three sections which is to teach the user the basic operation of the system. This is not to say that these options are in anyway more difficult to use, but rather that they are not required to successfully utilize the system in a meaningful manner. The basic purpose for all of these system options is to improve the ability of the user to tailor the report writing capability to his specific needs. These options are part of the SCHEDULE program but unless specifically enable by the SCHEDULE programmer will not appear as options during a SCHEDULE session. There are currently four options available the details of which are discussed herein.

4.5.2 Window

The window option allows the user to display in a report only the tasks which either occur or begin during a specific period of time. The time period which is of interest is entered into the program in reply to prompts which are displayed after the window option has been requested. This request is made early in the report writing procedure as shown here:

DO YOU WISH TO WRITE REPORTS OR TO EDIT DATA?

ENTER

E - EDIT DATA
R - WRITE REPORTS
N - NONE

R

YOU DO WANT A WINDOW OF DATA?

To enable the window option, the response should be yes:

DO YOU WANT A WINDOW OF DATA?

Y

The system will then prompt:

ENTER LOWER TIME LIMIT

/--*---*--/

which is a request for the earliest date to be included in the window. For this example, the response will be:

ENTER LOWER TIME LIMIT

/ 1*JAN*78/

The next prompt:

ENTER LOWER TIME LIMIT

/ 1*JAN*78/

ENTER UPPER TIME LIMIT

/--*---*--/

will ask for the latest date to be included in the window. For this example, the response is:

ENTER UPPER TIME LIMIT

/ 1*JAN*79/

At this point, the system will return to the normal report writing procedure:

ENTER UPPER TIME LIMIT

/ 1*JAN*79/
HOW MANY SYSTEMS DO YOU WANT TO SEE?
ENTER
A - ALL
S - SUBSET OF ALL
N - NONE

as covered in Section 3.0. At this point, the system has the required information to adjust the form as necessary and display the graph. The system will compute the number of years to be included in the report, based on the data specifying the window, hence, this question will not be asked. This will be the only deviation from the normal report writing steps. Figures 4.14 and 4.15 are a before and after example illustrating the affect of the window command based on the values entered within this example.

4.5.3 Lines Per Page

SCHEDULE has a default of 30 lines per page which means that when writing a report, a second page will not be required by the program until more than 30 items are entered. At some time, it may be necessary to change the default value to a lesser value; this may be accomplished through the lines per page option. If this option is enabled a prompt will appear during the normal report writing procedure as:

ENTER OFFICE PREPARING REPORTS
SECO
DO YOU WANT THE MAXIMUM LINES PER PAGE TO BE 30

A yes to this question will cause normal processing to continue as shown:

DO YOU WANT THE MAXIMUM LINES PER PAGE TO BE 30
Y
ENTER FILE CODE FOR SYSTEM
XXX0000 - FILE CODE
N - NONE / NO MORE SYSTEMS

However, a no response will cause the following prompt to be displayed:

```
ENTER OFFICE PREPARING REPORTS
SECO
DO YOU WANT THE MAXIMUM LINES PER PAGE TO BE 30
N
ENTER NEW MAXIMUM NUMBER OF LINES PER PAGES
```

The response to this query may not be greater than 30 nor less than nine. Figure 4.16 shows a report that was processed normally and Figure 4.17 shows the same report processed with the line per page option. Notice that the second report has spaced the items and is generally in a better format. Realizing from the first report that there were 35 items, it was logical to change the maximum number of lines per page to eighteen, hence:

```
ENTER NEW MAXIMUM NUMBER OF LINES PER PAGES
18
```

This is all the information required by the lines per page option and from this point, processing continues normally:

```
ENTER NEW MAXIMUM NUMBER OF LINES PER PAGES
18
ENTER FILE CODE FOR SYSTEM
XXX0000 - FILE CODE
N - NONE / NO MORE SYSTEMS
```

4.5.4 Application Code Prompting List

Different offices may use different definitions for various application code letters, thus requiring different prompting menus displayed to the user when writing a report. This option allows the user (or using office) to display a particular suggested list of application code definitions or to have no list displayed at all. The no-list option will appear in the normal processing stream as:

```
ENTER FILE CODE FOR SYSTEM
XXX0000 - FILE CODE
N - NONE / NO MORE SYSTEMS
001
ENTER(OFFICE/SUBJECT/REPORT/APLICODE) OF INTEREST
```

The Project Manager's Office suggested list will appear as:

ENTER(OFFICE/SUBJECT/REPORT/APLICODE) OF INTEREST

- A - ALL
- B - BUDGET
- E - PROJECT ENGINEER
- H - PROJECT MANAGER
- O - 104 REPORT
- P - PROCUREMENT
- Q - PRODUCT ASSURANCE
- R - RECAP
- S - PROGRAM SUMMARY
- T - TEST
- N - NONE

and the Research and Development Office suggested list will appear as:

ENTER(OFFICE/SUBJECT/REPORT/APLICODE) OF INTEREST

- A - ALL
- C - COMPTROLLER
- B - PLANS & ANALYSIS
- P - PROCUREMENT
- E - DEVELOPMENT & ENGINEERING
- F - ASTIO
- T - TRADOC
- S - TECOM
- Q - PRODUCT ASSURANCE
- N - NONE

which option appears depends on how the option is enabled. After the required application code has been entered, the processing will continue normally. It should be pointed out again that these are suggested lists and that the application codes eventually employed are up to the user.

4.5.5 Cross System

When enabled this option permits the user to write a single report covering all the systems in the master data file at one time. This should not be confused with the ALL option available when writing reports:

HOW MANY SYSTEMS DO YOU WANT TO SEE?

ENTER

A - ALL
S - SUBSET
N - NONE

which allows the user to write a separate report for all of the systems at the same time. The cross system reporting option allows the user to prepare one report which may contain all or part of the information from all the systems. Even when enabled this option only becomes available when the user requests to see all systems:

HOW MANY SYSTEMS DO YOU WANT TO SEE?

ENTER

A - ALL
S - SUBSET OF ALL
N - NONE

A

which will be followed by:

DO YOU WANT ALL SYSTEMS ON ONE REPORT?

A no response will cause the system to process the ALL request normally (i.e., one report for each system) and a yes answer will provide the cross system report.

The report output under this option may be quite lengthy in some cases and heading information for systems other than the first (system code 001) will not be part of the report. Because of this, it is wise to include in each system a dummy task or event as the first record (Record 0001) which identifies the system in the title task portion of the record. This may be given an application code such that it will only appear on the cross system report and not on the individual system report.

Appendix A

SCHEDULE Subroutine Listings

```

ADD: PROC(NLABEL,LINE);
DCL PROMPT CHAR(1) EXTERNAL;
DCL PROMPT1 CHAR(1);
DECLARE LINE CHAR(114);
DECLARE 1 CARD BASED(P),
    2 DELETE_FLAG BIT(8),
    2 ASE_CODE CHAR(7),
    2 APLI_CODE CHAR(11),
    2 MAJOR_SYSTEM1 CHAR(12),
    2 MAJOR_SYSTEM2 CHAR(20),
    2 TITLE1 CHAR(20),
    2 TITLE2 CHAR(24),
    2 AS_OF_DATE CHAR(10),
    2 NO_DVL_STPS FIXED DEC(5,0),
    2 NO_REMARKS FIXED DEC(5,0),
    2 FILE_END FIXED DEC(5,0);
DECLARE 1 DEVEL_STEP BASED(Q),
    2 DELETE_FLAG BIT(8),
    2 INFO,
    3 ASE_CODE CHAR(7),
    3 APLI_CODE CHAR(11),
    3 TYPE_NUMBER FIXED DEC(2),
    3 TITLE CHAR(28),
    3 STEP_DATES(9) FIXED DEC(5,2),
    3 COST(4) FIXED DEC(5,2),
    3 REST CHAR(26);
DECLARE 1 REMARKS BASED(R),
    2 DELETE_FLAG BIT(8),
    2 ASE_CODE CHAR(7),
    2 APLI_CODE CHAR(11),
    2 REMARK CHAR(80),
    2 REST CHAR(15);
DCL NLABEL FIXED BIN;
DCL TOUTPT ENTRY(FIXED BIN(31));
DCL INITT ENTRY(FIXED BIN(31));
DCL IX0X FIXED DEC(5,0);
DECLARE ANS CHAR(80) VAR;
DECLARE NUMBER CHAR(10) INIT('0123456789');
DECLARE TYPE CHAR(5) INIT('EVENT');
DCL START_DATE_CODE(5) FIXED DEC(5,2);
DCL STOP_DATE_CODE(5) FIXED DEC(5,2);
DCL EVENT_DATE_CODE(5) FIXED DEC(5,2);
DCL CAL_DATE FIXED DEC(5,2);
DECLARE (NTYPE,NUMB,STOPI) FIXED;
DCL RDATE ENTRY(CHAR(9),FIXED DEC(5,2));
DCL FDATE ENTRY(CHAR(9),FIXED DEC(5,2),FIXED BIN(15));
DCL STP_TITLE CHAR(28) VAR, ID_NO FIXED;

```

```

DCL COMPLET(10) FIXED DEC(5,2);
DCL (EVENT_DATE(10),START_DATE(10),STOP_DATE(10)) CHAR(9);
DCL READ#(11) LABEL;
DCL LABEL#(12) LABEL;
DCL (NREAD,I,II,III,NNN,MMM) FIXED BIN(5,0);
/* ***** */
P = ADDR(LINE);
Q = P;
R = P;
/* ***** */
ON CONVERSION BEGIN;
PUT EDIT('YOUR INPUT WAS IN ERROR') (COL(1), A);
PUT EDIT(ONSOURCE) (COL(1),A);
PUT EDIT('RE-ENTER') (COL(2), A);
PUT SKIP(2);
REVERT CONVERSION;
GO TO READ#(NREAD);
END;
GO TO LABEL#(NLABEL);
LABEL#(1)::
NREAD = 1;
READ#(1): PUT EDIT('ENTER MAJOR SYSTEMS') (COL(1),A);
PUT EDIT(' / _____ / _____ / ') (COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(MAJOR_SYSTEM1,MAJOR_SYSTEM2) (COL(2),A(12),X(1),A(20));
READ#(2): PUT EDIT('ENTER TITLE') (COL(1), A);
PUT EDIT(' / _____ / _____ / ')
(COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(TITLE1,TITLE2) (COL(2), A(20), X(1), A(24)) ;
NREAD = 3;
READ#(3): PUT EDIT('AS OF DATE') (COL(1), A);
PUT EDIT(' / __*__*__ / ') (COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(AS_OF_DATE) (COL(2), A(10)) ;
NREAD = 4;
READ#(4): PUT EDIT('ENTER START YEAR FOR REPORT') (COL(1),A);
PUT EDIT(' / ____ / ') (COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(NO_DVL_STPS) (COL(2), F(5,0)) ;
NREAD = 5;
READ#(5): PUT EDIT('ENTER NUMBER OF YEARS IN REPORT') (COL(1), A);
PUT EDIT(' / ____ / ') (COL(1),A);

```

```

PUT SKIP(2);
CALL REVERSE;
GET EDIT(NO_REMARKS) (COL(2), F(5,0)) ;
LABX0:PUT EDIT('APPLICATION CODE') (COL(2),A);
PUT EDIT(' / _____ /') (COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(CARD.APLI_CODE) (COL(2),A(11));
/*      *****      */
CALL NEWPAG;
CALL TSEND;
RETURN;
/*      *****      */
LABEL#(2)::
DO IX0X = 1 TO 9;
STEP_DATES(IX0X) = 0.0;
END;
DO IX0X = 1 TO 4;
COST(IX0X) = 0.0;
END;
SUBSTR(LINE,113,1) = 'S';
NREAD = 6;
READ#(6): PUT EDIT ('ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS')
(COL(1),A);
IF PROMPT = 'Y' | PROMPT1 = 'Y' THEN
PUT EDIT(' TASK OR EVENT ', ' FOLLOWED BY NUMBER')
(COL(2),A);
PUT SKIP(2);
GET EDIT(ANS) (COL(1), A(80)) ;
IF SUBSTR(ANS,1,1) = 'E' THEN NTYPE = 1;
ELSE NTYPE = 0;
NUMB = INDEX(ANS,' ') + 1;
NNN = 1;
NUMB = INDEX(NUMBER,SUBSTR(ANS,NUMB,1)) - 1;
IF NUMB > 0 THEN DO;
STOPI = NUMB;
TYPE_NUMBER = -STOPI ;
END;
ELSE DO;
STOPI = 1;
TYPE_NUMBER = -1;
END;
IF NTYPE = 0 THEN GO TO LAB_EV;
NREAD = 7;
READ#(7): PUT EDIT('          TITLE          / STR.DT. '
,' / STP.DT / COMPLT /') (COL(1), A, A);
PUT EDIT(' / _____ / * _ * _ /',

```


A-5

```

NNN = NNN + 1;
NREAD = 9;
READ#(9): PUT EDIT('/ EVENT DATE /') (COL(1), A);
PUT EDIT('/_*___*__/') (COL(1), A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(EVENT_DATE(NNN)) (COL(2), A(9));
CALL FDATE(EVENT_DATE(NNN), START_DATE_CODE(NNN), 1);
STEP_DATES(NNN) = START_DATE_CODE(NNN) - 1;
END;

/* ***** END INPUT OF AN EVENT ***** */
LAB_TE:;
/* ***** */
TITLE = STP_TITLE;
GO TO LABX0;
/* ***** */
LABEL#(3):;
NREAD = 11;
READ#(11): PUT EDIT('ENTER REMARKS')
(COL(1), A);
PUT EDIT('/ * _____ ,
' _____ ,
' _____ */') (COL(1), A, A, A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(REMARK) (COL(3), A(80));
SUBSTR(LINE, 113, 1) = 'R';
/* ***** */
GO TO LABX0;
/* ***** */
END ADDD;

```

```

CHANGE: PROC(NLABEL,LINE,FIN);
DCL PROMPT CHAR(1) EXTERNAL;
DCL PROMPT1 CHAR(1);
DCL FIN CHAR(1);
DECLARE LINE CHAR(114);
DECLARE 1 CARD BASED(P),
    2 DELETE_FLAG BIT(8),
    2 ASE_CODE CHAR(7),
    2 APLI_CODE CHAR(11),
    2 MAJOR_SYSTEM1 CHAR(12),
    2 MAJOR_SYSTEM2 CHAR(20),
    2 TITLE1 CHAR(20),
    2 TITLE2 CHAR(24),
    2 AS_OF_DATE CHAR(10),
    2 NO_DVL_STPS FIXED DEC(5,0),
    2 NO_REMARKS FIXED DEC(5,0),
    2 FILE_END FIXED DEC(5,0);
DECLARE 1 DEVEL_STEP BASED(Q),
    2 DELETE_FLAG BIT(8),
    2 INFO,
    3 ASE_CODE CHAR(7),
    3 APLI_CODE CHAR(11),
    3 TYPE_NUMBER FIXED DEC(2),
    3 TITLE CHAR(28),
    3 STEP_DATES(9) FIXED DEC(5,2),
    3 COST(4) FIXED DEC(5,2),
    3 REST CHAR(26);
DECLARE 1 REMARKS BASED(R),
    2 DELETE_FLAG BIT(8),
    2 ASE_CODE CHAR(7),
    2 APLI_CODE CHAR(11),
    2 REMARK CHAR(80),
    2 REST CHAR(15);
DCL NLABEL FIXED BIN;
DCL DUMMY FIXED DEC(5,2);
DECLARE ANS CHAR(80) VAR;
DCL ANS1 CHAR(80) INIT(' ');
DECLARE LETTER CHAR(27) INIT(' ABCDEFGHIJKLMNOPQRSTUVWXYZ');
DCL LETTER1 CHAR(26);
DCL START_DATE_CODE(5) FIXED DEC(5,2);
DCL STOP_DATE_CODE(5) FIXED DEC(5,2);
DCL INITT ENTRY(FIXED BIN(31));
DCL TOUTPT ENTRY(FIXED BIN(31));
DCL FDATE ENTRY(CHAR(9),FIXED DEC(5,2),FIXED BIN(15));
DCL RDATE ENTRY(CHAR(9),FIXED DEC(5,2));
DCL STP_TITLE CHAR(28) VAR;
DCL COMPLET(10) FIXED DEC(5,2);

```



```

DCL (EVENT_DATE(10),START_DATE(10),STOP_DATE(10)) CHAR(9);
DCL READ#(13) LABEL;
DCL LABEL#(13) LABEL;
DCL (NREAD,I,NNN,STOPI,STOPII) FIXED BIN;
DCL NOOVLSTPS FIXED DEC(5,0);
DCL NOREMARKS FIXED DEC(5,0);
LETTER1 = SUBSTR(LETTER,2,26);
/* ***** */
P = ADDR(LINE);
Q = P;
R = P;
/* ***** */
ON CONVERSION BEGIN;
PUT EDIT('YOUR INPUT WAS IN ERROR') (COL(1), A);
PUT EDIT(ONSOURCE) (COL(1), A);
PUT EDIT('RE-ENTER') (COL(2), A);
PUT SKIP(2);
REVERT CONVERSION;
GO TO READ#(NREAD);
END;
GO TO LABEL#(NLABEL);
LABEL#(1):;
NREAD = 1;
READ#(1): PUT EDIT('ENTER MAJOR SYSTEMS') (COL(1), A);
PUT EDIT(MAJOR_SYSTEM1,MAJOR_SYSTEM2) (COL(2), A,X(1), A);
PUT EDIT('/_____/_____/') (COL(1), A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1), A(80));
IF SUBSTR(ANS,1,1) = 'N' | SUBSTR(ANS,1,1) = 'S'
THEN GO TO LABEL#(13);
IF SUBSTR(ANS,2,12) ~= SUBSTR(ANS1,1,12) THEN
MAJOR_SYSTEM1 = SUBSTR(ANS,2,12);
IF SUBSTR(ANS,15,20) ~= SUBSTR(ANS1,1,20) THEN
MAJOR_SYSTEM2 = SUBSTR(ANS,15,20);
/* ***** */
GO TO LABEL#(13);
/* ***** */
LABEL#(2):;
NREAD = 2;
READ#(2): PUT EDIT('ENTER TITLE') (COL(1), A);
PUT EDIT(TITLE1,TITLE2) (COL(2), A,X(1), A);
PUT EDIT('/_____/_____/')
(COL(1), A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1), A(80));

```



```

IF SUBSTR(ANS,1,1) = 'N' I SUBSTR(ANS,1,1) = 'S'
THEN GO TO LABEL#(13);
IF SUBSTR(ANS,2,20) ^= SUBSTR(ANS1,1,20) THEN
TITLE1 = SUBSTR(ANS,2,20);
IF SUBSTR(ANS,23,24) ^= SUBSTR(ANS1,1,24) THEN
TITLE2 = SUBSTR(ANS,23,24);
/* ***** */
GO TO LABEL#(13);
/* ***** */
LABEL#(3):;
NREAD = 3;
READ#(3): PUT EDIT('AS OF DATE') (COL(1), A);
PUT EDIT(AS_OF_DATE) (COL(2), A);
PUT EDIT('/_*_*_/_') (COL(1), A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1), A(80));
IF SUBSTR(ANS,1,1) = 'N' I SUBSTR(ANS,1,1) = 'S'
THEN GO TO LABEL#(13);
IF SUBSTR(ANS,2,10) ^= SUBSTR(ANS1,1,10) THEN
AS_OF_DATE = SUBSTR(ANS,2,10);
/* ***** */
GO TO LABEL#(13);
/* ***** */
LABEL#(4):;
NREAD = 4;
READ#(4): PUT EDIT('ENTER START YEAR FOR REPORT') (COL(1), A);
PUT EDIT(NO_DVL_STPS) (COL(2), F(5,0));
PUT EDIT('/_/_/_/_/_') (COL(1), A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1), A(80));
IF SUBSTR(ANS,1,1) = 'N' I SUBSTR(ANS,1,1) = 'S'
THEN GO TO LABEL#(13);
GET STRING(ANS) EDIT(NODVLSTPS) (X(1), F(5,0)) ;
IF NODVLSTPS ^=0 THEN NO_DVL_STPS = NODVLSTPS;
/* ***** */
GO TO LABEL#(13);
/* ***** */
LABEL#(5):;
NREAD = 5;
READ#(5): PUT EDIT('ENTER NUMBER OF YEARS IN REPORT') (COL(1), A);
PUT EDIT(NO_REMARKS) (COL(2), F(5,0));
PUT EDIT('/_/_/_/_/_') (COL(1), A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1), A(80));

```

```

IF SUBSTR(ANS,1,1) = 'N' | SUBSTR(ANS,1,1) = 'S'
THEN GO TO LABEL#(13);
GET STRING(ANS) EDIT(NOREMARKS) (X(1), F(5,0)) ;
IF NOREMARKS ^=0 THEN NO_REMARKS = NOREMARKS ;
/* ***** */
GO TO LABEL#(13);
/* ***** */
LABEL#(6)::
NREAD = 6;
PUT EDIT(CARD.APLI_CODE) (COL(2),A);
PUT EDIT('/_____/') (COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1),A(80));
IF SUBSTR(ANS,1,1) = 'N' | SUBSTR(ANS,1,1) = 'S'
THEN GO TO LABEL#(13);
IF SUBSTR(ANS,2,11) ^= SUBSTR(ANS1,1,11) THEN
CARD.APLI_CODE = SUBSTR(ANS,2,11);
/* ***** */
GO TO LABEL#(13);
/* ***** */
LABEL#(7)::
NREAD = 7;
READ#(7): PUT EDIT('          TASK          / ')
(COL(1),A);
PUT EDIT(DEVEL_STEP.TITLE) (COL(2),A);
PUT SKIP(2);
PUT EDIT('/_____/') (COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1),A(80));
IF SUBSTR(ANS,1,1) = 'N' | SUBSTR(ANS,1,1) = 'S'
THEN GO TO LABEL#(13);
IF SUBSTR(ANS,2,28) ^= SUBSTR(ANS1,1,28) THEN
DEVEL_STEP.TITLE = SUBSTR(ANS,2,28);
/* ***** */
GO TO LABEL#(13);
/* ***** */
LABEL#(8)::
READ#(8): NREAD = 8;
STOPII = Q->TYPE_NUMBER;
STOPII = ABS(STOPII);
NNN = 0;
MMM = -2;
DO I = 1 TO STOPII;
MMM = MMM + 3;
NNN = NNN + 1;

```

```

PUT EDIT(' / STR.DT. / STP.DT. / COMPT /')
(COL(1), A);
DUMMY = STEP_DATES(MMM) + 1;
CALL RDATE(START_DATE(NNN), DUMMY);
DUMMY = STEP_DATES(MMM+1) + 1;
CALL RDATE(STOP_DATE(NNN), DUMMY);
COMPLET(NNN) = STEP_DATES(MMM+2);
PUT EDIT(START_DATE(NNN), STOP_DATE(NNN), COMPLET(NNN))
(COL(2), A, X(1), A, X(2), F(5,2));
PUT EDIT(' / _ * _ _ * _ / _ * _ _ * _ / *** _ _ ** /') (COL(1), A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1), A(80));
IF SUBSTR(ANS, 1, 1) = 'N' | SUBSTR(ANS, 1, 1) = 'S'
THEN GO TO LABEL#(13);
GET STRING(ANS) EDIT(START_DATE(NNN), STOP_DATE(NNN), COMPLET(NNN))
(X(1), A(9), X(1), A(9), X(4), F(5,2));
IF COMPLET(NNN) /= 0 THEN STEP_DATES(MMM+2) = COMPLET(NNN);
IF VERIFY(START_DATE(NNN), LETTER) /= 0 THEN DO;
CALL FDATE(START_DATE(NNN), DUMMY, 2);
STEP_DATES(MMM) = DUMMY - 1;
END;
IF VERIFY(STOP_DATE(NNN), LETTER) /= 0 THEN DO;
CALL FDATE(STOP_DATE(NNN), DUMMY, 3);
STEP_DATES(MMM+1) = DUMMY - 1;
END;
END;
/* ***** */
GO TO LABEL#(13);
/* ***** */
LABEL#(9);
READ#(9): PUT EDIT(' / EVENT /')
(COL(1), A);
PUT EDIT(DEVEL_STEP.TITLE)
(COL(2), A);
PUT EDIT(' / _____ /')
(COL(1), A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1), A(80));
IF SUBSTR(ANS, 1, 1) = 'N' | SUBSTR(ANS, 1, 1) = 'S'
THEN GO TO LABEL#(13);
IF SUBSTR(ANS, 2, 28) /= SUBSTR(ANS1, 1, 28) THEN
DEVEL_STEP.TITLE = SUBSTR(ANS, 2, 28);
/* ***** */
GO TO LABEL#(13);
/* ***** */

```



```

LABEL#(10)::
READ#(10): NREAD = 10;
STOPII = Q->TYPE_NUMBER;
STOPII = ABS(STOPII);
NNN = 0;
DO I = 1 TO STOPII;
MMM = NNN + 1;
NNN = NNN + 1;
PUT EDIT('/ EVENT DATE /') (COL(1), A);
DUMMY = STEP_DATES(NNN) + 1;
CALL RDATE(EVENT_DATE(NNN), DUMMY);
PUT EDIT(EVENT_DATE(NNN)) (COL(2), A);
PUT EDIT('/_*___*/') (COL(1), A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1), A(80));
IF SUBSTR(ANS, 1, 1) = 'N' | SUBSTR(ANS, 1, 1) = 'S'
THEN GO TO LABEL#(13);
GET STRING(ANS) EDIT(EVENT_DATE(NNN)) (X(1), A(9)) ;
IF VERIFY(EVENT_DATE(NNN), LETTER) ~= 0 THEN DO;
CALL FDATE(EVENT_DATE(NNN), DUMMY, 1);
STEP_DATES(MMM) = DUMMY - 1;
END;
END;
/* ***** */
GO TO LABEL#(13);
/* ***** */
LABEL#(11)::
NREAD = 11;
READ#(11): PUT EDIT('ENTER REMARKS')
(COL(1), A);
PUT EDIT(REMARK) (COL(2), A);
PUT EDIT('/*_-----',
'-----',
'-----*/') (COL(1), A, A, A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1), A(80));
IF SUBSTR(ANS, 1, 1) = 'N' | SUBSTR(ANS, 1, 1) = 'S'
THEN GO TO LABEL#(13);
IF SUBSTR(ANS, 3, 77) ~= SUBSTR(ANS, 1, 80) THEN
REMARK = SUBSTR(ANS, 3, 77);
/* ***** */
GO TO LABEL#(13);
/* ***** */
LABEL#(12)::
PUT EDIT('THE ELEMENT TITLE IS ... ', DEVEL_STEP.TITLE)

```



```

(COL(2),A,A);
PUT EDIT('ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS')
(COL(1),A);
IF PROMPT = 'Y' | PROMPT1 = 'Y' THEN DO;
PUT EDIT(' TASK OR EVENT ', ' FOLLOWED BY NUMBER')
(COL(2),A);
PROMPT1 = 'N';
END;
PUT SKIP(2);
GET EDIT(ANS) (COL(1),A(80));
IF SUBSTR(ANS,1,1) = 'N' THEN GO TO LABEL#(13);
NUMB = VERIFY(ANS,LETTER);
IF NUMB > 0 THEN GET STRING(ANS) EDIT(STOPI) (X(NUMB-1),F(1));
ELSE STOPI = 1;
NNN = 0;
IF SUBSTR(ANS,1,1) = 'E' THEN GO TO LAB_EV;
IF STOPI > 3 THEN DO;
PUT EDIT('MAX NUMBER OF SUB TASKS IS 3')
(COL(2),A);
PUT SKIP(2);
PROMPT1 = 'Y';
GO TO LABEL#(12);
END;
DO I = 1 TO STOPI;
NNN = NNN + 1;
MMM = 3*(NNN-1) + 1;
NREAD = 12;
READ#(12): PUT EDIT('/ STR.DT. / STP.DT. / COMPLT /')
(COL(1), A);
PUT EDIT('/_*_*_*_/_*_*_*_/***_*_*_/') (COL(1), A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1),A(80));
IF SUBSTR(ANS,1,1) = 'N' THEN GO TO LABEL#(13);
GET STRING(ANS) EDIT(START_DATE(NNN),STOP_DATE(NNN),COMPLET(NNN))
(X(1),A(9),X(1),A(9),X(4),F(5,2)) ;
CALL FDATE(START_DATE(NNN),START_DATE_CODE(NNN),2);
STEP_DATES(MMM) = START_DATE_CODE(NNN)-1;
CALL FDATE(STOP_DATE(NNN),STOP_DATE_CODE(NNN),3);
STEP_DATES(MMM+1) = STOP_DATE_CODE(NNN) -1;
STEP_DATES(MMM+2) = COMPLET(NNN);
STOPII = NNN;
END;
TYPE_NUMBER = -STOPII;
/* ***** */
GO TO LABEL#(13);
/* ***** */

```

```

LAB_EV::
DO I = 1 TO STOPI;
NNN = NNN + 1;
NREAD = 13;
READ#(13): PUT EDIT(' / EVENT DATE / ') (COL(1), A);
PUT EDIT(' / ____*____*____/ ') (COL(1), A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1), A(80));
IF SUBSTR(ANS,1,1) = 'N' THEN GO TO LABEL#(13);
GET STRING(ANS) EDIT(EVENT_DATE(NNN)) (X(1), A(9)) ;
CALL FOATE(EVENT_DATE(NNN), START_DATE_CODE(NNN), 1);
STEP_DATES(NNN) = START_DATE_CODE(NNN) - 1;
STOPII = NNN;
END;
TYPE_NUMBER = STOPII;
LABEL#(13):: /* NO CHANGES */
FIN = SUBSTR(ANS,1,1) ;
IF FIN = 'S' THEN DO;
IF NLABEL = 8 | NLABEL = 10 THEN GO TO LABZZX;
FIN = 'N';
END;
LABZZX::
CALL NEWPAGE;
CALL TSEND;
/* ***** */
RETURN;
/* ***** */
END CHANGE;

```

```

FDATE: PROC(DAY_MONTH_YEAR,FDATE_CODE,NDTYPE);
DCL DTYPE(0:4) CHAR(6) INIT(' ','EVENT','START',' STOP');
DCL NDTYPE FIXED BIN(15);
DCL DMC PIC'99AAAAA99';
DCL A1 CHAR(1);
DCL INITT ENTRY(FIXED BIN(31));
DCL TOUTPT ENTRY(FIXED BIN(31));
DECLARE (L,I,J,K) FIXED;
DECLARE DAY_MONTH_YEAR CHAR(10);
DECLARE (MONTH) CHARACTER(3);
DCL (YEAR,MONTH_CODE) PIC'999V99';
DECLARE DAY_FACTOR(3,4) FIXED DEC(5,2)
INIT(31.0,30.0,31.0,31.0,
28.0,31.0,31.0,30.0,
31.0,30.0,30.0,31.0);
DECLARE FDATE(3,4) CHARACTER(3)
INIT('JAN','APR','JUL','OCT',
'FEB','MAY','AUG','NOV',
'MAR','JUN','SEP','DEC');
DECLARE START_YEAR FIXED DEC(5,2) INIT(76.0),
FDATE_CODE FIXED DEC(5,2),DAY PIC'999V99';
ON CONVERSION BEGIN;
PUT EDIT('      YOUR DATE CONTAINED AN ERROR ',
'IT MUST BE CHANGED TO ',
'ALLOW PROCESSING TO CONTINUE      ')
(COL(5),A,A,A);
PUT EDIT(ONSOURCE) (COL(1),A);
PUT EDIT('/_#___#___/')(COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(A1,DMC) (COL(1),A(1),P'99AAAAA99');
DAY_MONTH_YEAR = DMC;
GO TO JBA;
END;
BEGIN;
IF SUBSTR(DAY_MONTH_YEAR,3,1) = ' ' |
SUBSTR(DAY_MONTH_YEAR,7,1) = ' ' THEN DO;
PUT EDIT(DTYPE(NDTYPE),' DATE IN ERROR YOU MUST RE-ENTER ')
(COL(1),(2)A);
PUT EDIT(DAY_MONTH_YEAR) (COL(2),A);
PUT EDIT('/_#___#___/')(COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(DAY_MONTH_YEAR) (COL(2),P'99AAAAA99');
END;
END;
JBA: DAY = SUBSTR(DAY_MONTH_YEAR,1,2);

```



```

MONTH = SUBSTR(DAY_MONTH_YEAR,4,3);
YEAR = SUBSTR(DAY_MONTH_YEAR,8,2);
/* ***** START FINDING MONTH CODE ***** */
FIX_MON: K=0; L=0; MONTH_CODE=0;
DO I=1 TO 3 ;
L= SUM(FDATE(I,*)=MONTH);
IF L=1 THEN DO J=1 TO 4 ;
K=SUM(FDATE(*,J)=MONTH);
IF K=1 THEN DO;
MONTH_CODE= 3*(J-1) + 1;
GO TO S600;
END;
END;
/* ***** END FINDING MONTH CODE ***** */
END;
PUT EDIT(' I CANNOT FIND YOUR MONTH      ',
' PLEASE RE-ENTER.')
```

(COL(5),A);

```

PUT EDIT(' /__/' ) (COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(MONTH) (COL(2),A(3));
GO TO FIX_MON;
/* ***** CALCULATE FDATE CODE ***** */
S600: FDATE_CODE = DAY / DAY_FACTOR(I,J)
+ MONTH_CODE
+ (YEAR-START_YEAR)*12 + .01;
/* ***** ***** */
S700: RETURN;
END FDATE;
```


AD-A072 140

ARMY AVIATION RESEARCH AND DEVELOPMENT COMMAND ST LO--ETC F/G 9/2
SCHEDULE. MANAGERIAL TOOL FOR PROJECT/PRODUCT MANAGERS.(U)
MAY 79 V ALLEN, L ANTWILER, M GARSIK

UNCLASSIFIED

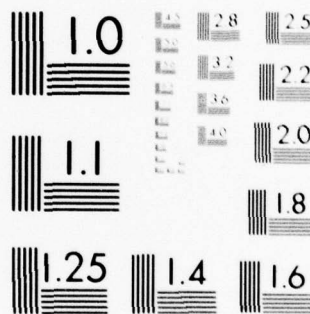
USAAVRADCOM-TR-79-17

NL

2 OF 3

AD
A072140





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```

GRAP: PROC(TYPE);
DCL ANS CHAR(80) VAR;
DCL (LLIMIT,ULIMIT,STYR,NOYR) FIXED DEC(5,2) EXTERNAL;
DCL PROMPT CHAR(1) EXTERNAL;
DCL (TYPE,RTYPE) CHAR(1);
DCL TOUTPT ENTRY(FIXED BIN(31));
DCL INITT ENTRY(FIXED BIN(31));
DCL SCRATCH FILE ENV(F(80,80));
DCL DIREC FILE RECORD KEYED ENV(INDEXED F(1140,114));
IF TYPE = 'N' THEN CALL NEW1;
IF TYPE = 'L' THEN CALL WINDOW;
IF TYPE = 'E' THEN CALL UPDATE;
IF TYPE = 'R' THEN DO;
PUT EDIT('ENTER TERMINAL TYPE') (COL(1),A);
IF PROMPT = 'Y' THEN
PUT EDIT('ENTER',
'G - TEXTRONIX 4014 (GRAPHICAL OUTPUT)',
'P - TEXAS INSTRUMENTS (PRINTER OUTPUT)',
'N - NONE')
(COL(1),A,(3)(COL(5),A));
PUT SKIP(2);
GET EDIT(RTYPE) (COL(1),A(1));
IF RTYPE = 'G' THEN CALL FORM;
IF RTYPE = 'P' THEN CALL TYREPT;
END;
RETURN;
END GRAP;

```

```

LLIST: PROC(LINE);
DECLARE LINE CHAR(114);
DECLARE 1 CARD BASED(P),
  2 DELETE_FLAG BIT(8),
  2 ASEC CHAR(7),
  2 APLI_CODE CHAR(11),
  2 MAJOR_SYSTEM1 CHAR(12),
  2 MAJOR_SYSTEM2 CHAR(20),
  2 TITLE1 CHAR(24),
  2 TITLE2 CHAR(20),
  2 AS_OF_DATE CHAR(10),
  2 NO_DVL_STPS FIXED DEC(5,0),
  2 NO_REMARKS FIXED DEC(5,0),
  2 FILE_END FIXED DEC(5,0);
DECLARE 1 DEVEL_STEP BASED(Q),
  2 DELETE_FLAG BIT(8),
  2 INFO,
  3 ASE_CODE CHAR(7),
  3 APLI_CODE CHAR(11),
  3 TYPE_NUMBER FIXED DEC(2),
  3 TITLE CHAR(28),
  3 STEP_DATES(9) FIXED DEC(5,2),
  3 COST(4) FIXED DEC(5,2),
  3 REST CHAR(26);
DECLARE 1 REMARKS BASED(R),
  2 DELETE_FLAG BIT(8),
  2 ASE_CODE CHAR(7),
  2 APLI_CODE CHAR(11),
  2 REMARK CHAR(80),
  2 REST CHAR(15);
DCL ALDATE(9) CHAR(9);
DCL APLI CHAR(11) VAR;
DCL DUMMY FIXED DEC(5,2);
/* ***** */
P = ADDR(LINE);
Q = P;
R = P;
/* ***** */
IF SUBSTR(LINE,5,4) = '0000' THEN DO;
PUT EDIT('SYSTEM NUMBER: ',SUBSTR(ASEC,1,3))
(COL(1),A,A);
PUT EDIT('MAJOR SYSTEM: ',MAJOR_SYSTEM1,MAJOR_SYSTEM2) (COL(1),A,A,A);
PUT EDIT('PROJECT TITLE: ',TITLE1,TITLE2) (COL(1),A,A,A,A);
PUT EDIT('AS OF DATE: ',AS_OF_DATE) (COL(1),A,A);
PUT SKIP(2);
RETURN;
END;

```



```

APLI = CARD.APLI_CODE;
LSTR1 = VERIFY(APLI,' ');
IF LSTR1 = 0 | LSTR1 = 11 THEN DO;
NSP = 0;
GO TO XX;
END;
LSTR2 = INDEX(APLI,' ');
DO I = 11 TO 1 BY -1;
IF INDEX(SUBSTR(APLI,I,1),' ') = 0 THEN DO;
NSP = -LSTR1 + I;
GO TO XX;
END;
END;
XX: APLI = SUBSTR(APLI,LSTR1,NSP+1);
NSP = 10-NSP;
IF SUBSTR(LINE,113,1) = 'S' THEN DO;
IF TYPE_NUMBER > 0 THEN DO;
LE: DO K = 1 TO TYPE_NUMBER;
DUMMY = STEP_DATES(K) + 1;
CALL RDATE(ALDATE(K),DUMMY);
END LE;
PUT EDIT(SUBSTR(ASEC,4,4),'( ',APLI,') ',TITLE,
(ALDATE(K) DO K = 1 TO TYPE_NUMBER))
(COL(1),A,X(1),X(NSP),A,A,A,COL(20),A,X(1),
(TYPE_NUMBER)(COL(49),A,X(1),A));
END;
NUMBER = -TYPE_NUMBER;
IF TYPE_NUMBER < 0 THEN DO;
LS: DO KK = 1,2,4,5,7,8 WHILE(TYPE_NUMBER*3<-KK);
DUMMY = STEP_DATES(KK) + 1;
CALL RDATE(ALDATE(KK),DUMMY);
END LS;
PUT EDIT(SUBSTR(ASEC,4,4),'( ',APLI,') ',TITLE,
(ALDATE(K),ALDATE(K+1),STEP_DATES(K+2)
DO K = 1 TO NUMBER*3 BY 3))
(COL(1),A,X(1),X(NSP),A,A,A,X(1),A,X(1),
(NUMBER)(COL(49),A,X(1),A,X(3),P'9V99'));
END;
PUT SKIP(2);
RETURN;
END;
IF SUBSTR(LINE,113,1) = 'R' THEN DO;
LSTR1 = INDEX(SUBSTR(REMARK,32,8),' ') + 31;
LSTR2 = VERIFY(SUBSTR(REMARK,LSTR1,80-LSTR1),' ');
PUT EDIT(SUBSTR(ASEC,4,4),'( ',APLI,') ',SUBSTR(REMARK,1,LSTR1))
(COL(1),A,X(1),X(NSP),A,A,A,COL(20),A);
IF LSTR2 = 0 THEN

```

PUT EDIT(SUBSTR(REMARK,LSTR1+1,79-LSTR1))
(COL(20),A);
PUT SKIP(2);
RETURN;
END;
RETURN;
END LLIST;

```

MASTER: PROCEDURE(JBACBA) OPTIONS(MAIN);
DCL (NSARY,NRARY) FIXED DEC(5) EXTERNAL;
DCL (LLIMIT,ULIMIT,STYR,NOYP) FIXED DEC(5,2) EXTERNAL;
DCL NSANS CHAR(1);
DCL JBACBA CHAR(100) VAR;
DCL TOUTPT ENTRY(FIXED BIN(31));
DCL INITT ENTRY(FIXED BIN(31));
DCL CHER FIXED DEC(5,2);
DECLARE LINE CHAR(114);
DCL (JENNY1,JENNY2,JENNY3,JENNY4,JENNY5) CHAR(1) EXTERNAL;
DCL ANS CHAR(80) VAR;
DCL PROMPT CHAR(1) EXTERNAL;
DCL TYPE CHAR(1);
DCL IX FIXED DEC(5,0);
DCL SCRATCH FILE ENV(F(80,80));
DCL DIREC FILE RECORD KEYED ENV(INDEXED F(1140,114));
CALL INITT(120);
CALL TSEND;
ULIMIT = 999.99;
IX = 0;
/*
JENNY1 = P => PROJECT MANAGER
JENNY1 = R => R & D COMMAND
JENNY1 = N => OTHER
JENNY2 = C => WRITE ACROSS SYSTEMS
JENNY2 = N => NOT AVAILABLE
JENNY3 = P => APPLICATION CODE POSITION DEPENDENT
JENNY3 = N => NOT AVAILABLE
JENNY4 = M => MAX LINES CONTROL
JENNY4 = N => NOT AVAILABLE
JENNY5 = W => WINDOW
JENNY5 = N => NOT AVAILABLE
*/
IF LENGTH(JBACBA) = 0 THEN DO;
JENNY1 = 'P';
JENNY2 = 'N';
JENNY3 = 'N';
JENNY4 = 'N';
JENNY5 = 'N';
NSARY = 1000;
NRARY = 100;
END;
ELSE DO;
JENNY1 = SUBSTR(JBACBA,1,1);
JENNY2 = SUBSTR(JBACBA,2,1);
JENNY3 = SUBSTR(JBACBA,3,1);
JENNY4 = SUBSTR(JBACBA,4,1);

```

```

JENNY5 = SUBSTR(JBACBA,5,1);
IF LENGTH(JBACBA) >= 10 THEN
GET STRING(JBACBA) EDIT(NSARY) (X(5),F(5));
ELSE NSARY = 1000;
IF LENGTH(JBACBA) >= 15 THEN
GET STRING(JBACBA) EDIT(NRARY) (X(10),F(5));
ELSE NRARY = 100;
END;
PUT EDIT('DO YOU NEED FULL PROMPTING ?') (COL(1),A);
PUT SKIP(2);
GET EDIT(PROMPT) (COL(1),A(5));
LABZ: PUT EDIT('ENTER SYSTEM TYPE') (COL(1),A);
IF PROMPT = 'Y' THEN
PUT EDIT('N - NEW (THE SYSTEM DOES NOT HAVE A CODE)',
'O - OLD (THE SYSTEM IS IN THE MAIN FILE)')
(COL(5),A);
PUT SKIP(2);
GET EDIT(TYPE) (COL(1),A(5));
IF TYPE = 'N' THEN CALL GRAP(TYPE);
IF TYPE = 'O' THEN DO;
LABZZ: PUT EDIT('DO YOU WISH TO WRITE REPORTS OR TO EDIT DATA?')
(COL(1),A);
IF PROMPT = 'Y' THEN
PUT EDIT('ENTER',
'E - EDIT DATA',
'R - WRITE REPORTS',
'N - NONE')
(COL(1),A,COL(5),A,COL(5),A,COL(5),A);
PUT SKIP(2);
GET EDIT(TYPE) (COL(1),A(1));
IF TYPE = 'N' THEN GO TO LABZZZ;
IF TYPE = 'E' THEN CALL GRAP(TYPE);
IF TYPE = 'R' THEN DO;
IF JENNY5 = 'W' THEN DO;
PUT EDIT('DO YOU WANT A WINDOW OF DATA ?') (COL(1),A);
PUT SKIP(2);
GET EDIT(NSANS) (COL(1),A(1));
IF NSANS = 'Y' THEN DO;
TYPE = 'L';
CALL GRAP(TYPE);
END;
ELSE DO;
LLIMIT = 0.0;
ULIMIT = 999.99;
END;
END;
CALL REPORTS(IX,TYPE);

```



```

TYPE = 'R';
IF IX = 1 THEN DO;
OPEN FILE(SCRATCH) INPUT;
CALL GRAP(TYPE);
CLOSE FILE(SCRATCH);
END;
ELSE;
END;
GO TO LABZZ;
END;
LABZZ: PUT EDIT('DO YOU WISH TO CONTINUE ?') (COL(1),A);
PUT SKIP(2);
GET EDIT(ANS) (COL(1),A(80));
IF ANS = 'Y' | ANS = 'YES' THEN GO TO LABZ;
END MASTER;

```

```

NEW: PROC(NO_SYS);
DCL PROMPT1 CHAR(1);
DCL PROMPT CHAR(1) EXTERNAL;
DCL RE_START CHAR(2) VAR;
DECLARE MAJOR_SYSTEM1 CHAR(12), MAJOR_SYSTEM2 CHAR(20);
DECLARE T_SEND CHAR(40)
INIT('          SCHEDULES          ');
DECLARE ANS CHAR(80) VAR;
DCL STOP1 FIXED DEC(1);
DCL REMARK CHAR(72) VAR;
DECLARE LETTER CHAR(27) INIT('ABCDEFGHIJKLMNOPQRSTUVWXYZ ');
DCL START_DATE_CODE(5) FIXED DEC(5,2);
DCL STOP_DATE_CODE(5) FIXED DEC(5,2);
DCL EVENT_DATE_CODE(5) FIXED DEC(5,2);
DCL CAL_DATE FIXED DEC(5,2);
DECLARE (NTYPE,NUMB) FIXED;
DCL INITT ENTRY(FIXED BIN(31));
DCL TOUTPT ENTRY(FIXED BIN(31));
DCL IOWAIT ENTRY(FIXED BIN(31));
DCL FDATE ENTRY(CHAR(9),FIXED DEC(5,2),FIXED BIN(15));
DCL TITLE1 CHAR(20) VAR, TITLE2 CHAR(24) VAR,
STP_TITLE CHAR(28) VAR, ID_NO FIXED,
AS_OF_DATE CHAR(9);
DCL (NO_DVP_STPS,NO_REMARKS,COMPLET(10)) FIXED DEC(5,2);
DCL (EVENT_DATE(10),START_DATE(10),STOP_DATE(10)) CHAR(9);
DCL NO_SYS FIXED DEC(5,0);
DCL NOSYS FIXED DEC(5,0);
DCL JBA CHAR(3) VAR;
DCL READ#(10) LABEL;
DCL (NREAD,I,II,III,NNN) FIXED DEC(5,0);
DCL STD FILE ENV(F(80,80));
DCL SCRATCH FILE ENV(F(80,80));
DCL MAXNO FIXED DEC(5,0);
OPEN FILE(STD) INPUT;
DCL STDTLE CHAR(28) BASED(TP);
DCL TIP(500) POINTER;
ON CONVERSION BEGIN;
PUT EDIT('YOUR INPUT WAS IN ERROR') (COL(1), A);
PUT SKIP(2);
PUT EDIT(ONSOURCE) (COL(1),A);
PUT SKIP(2);
PUT EDIT('RE-ENTER') (COL(2), A);
PUT SKIP(2);
REVERT CONVERSION;
GO TO READ#(NREAD);
END;
ON ENDFILE (STD) GO TO xx;

```

```

STLOOP: DO ID_NO = 1 TO 500;
ALLOCATE STDTLE;
GET FILE(STD) EDIT(STDTLE) (COL(1),A(28));
TIP(ID_NO) = TP;
END STLOOP;
XX: MAXNO = ID_NO - 1;
CLOSE FILE(STD);
PUT EDIT(' IS THIS A RE-START OF AN OLD JOB ?')
(COL(5),A);
PUT SKIP(2);
GET EDIT(RE_START) (COL(1),A(2));
IF SUBSTR(RE_START,1,1) = 'Y' THEN RESTART: DO;
BEGIN;
DCL HCARD CHAR(80) BASED(PC);
DCL PCARD(500) POINTER;
DCL HOLD CHAR(80) VAR;
DCL (I,J) FIXED DEC(5,0);
ON ENDFILE(SCRATCH) BEGIN;
IF I < 7 THEN DO;
PUT EDIT('RESTART FAILED          YOU MUST ENTER ALL DATA AGAIN')
(COL(5),A); PUT SKIP(2);
CLOSE FILE(SCRATCH);
GO TO XX4;
END;
GO TO XX1;
END;
OPEN FILE(SCRATCH) INPUT;
L1: DO I = 1 TO 500;
ALLOCATE HCARD;
GET FILE(SCRATCH) EDIT(HCARD) (COL(1),A(80));
PCARD(I) = PC;
END L1;
XX1: CLOSE FILE(SCRATCH);
OPEN FILE(SCRATCH) OUTPUT;
L2: DO J = 1 TO I-2;
PC = PCARD(J);
PUT FILE(SCRATCH) EDIT(HCARD) (COL(1),A);
IF J = 4 THEN FREE HCARD;
END L2;
PC = PCARD(4);
GET STRING(HCARD) EDIT(NO_DVP_STPS,NO_REMARKS) ((2)F(5,0));
FREE HCARD;
PC = PCARD(I-1);
NO_DVP_STPS = NO_DVP_STPS - I + 5;
IF NO_DVP_STPS > 0 THEN DO;
CALL BELL;
CALL TSEND;

```



```

PUT EDIT('THE LAST DEVELOPMENT STEP WAS') (COL(5),A);
PUT EDIT(HCARD) (COL(1),A);
PUT EDIT(' START WITH NEXT DEVELOPMENT STEP')
(COL(5),A);
PUT SKIP(2);
CALL IOWAIT(30);
GO TO XX2;
END;
IF NO_REMARKS + NO_DVP_STPS > 0 THEN DO;
CALL BELL;
CALL TSEND;
PUT EDIT('THE LAST REMARK WAS') (COL(5),A);
PUT EDIT(HCARD) (COL(1),A);
PUT EDIT('START WITH NEXT REMARK')
(COL(6),A);
PUT SKIP(2);
CALL IOWAIT(30);
GO TO NULLDVL;
END;
GO TO XX3;
END RESTART;
XX4: OPEN FILE(SCRATCH) OUTPUT;
/*      */
/* ***** START SET UP OF REPORT HEADING ***** */
NREAD = 1;
NREAD = 2;
CALL NEWPAGE;
CALL TSEND;
PUT EDIT('ENTER MAJOR SYSTEMS') (COL(1),A);
PUT EDIT('/_____/_____/') (COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(MAJOR_SYSTEM1,MAJOR_SYSTEM2) (COL(2),A(12),X(1),A(20));
READ#(2): PUT EDIT('ENTER PROJECT TITLE') (COL(1), A);
PUT EDIT('/_____/_____/')
(COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(TITLE1,TITLE2) (COL(2), A(20), X(1), A(24)) ;
NREAD = 3;
READ#(3): PUT EDIT('AS OF DATE') (COL(1), A);
PUT EDIT('/__*__*_/') (COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1),A(80));
GET STRING(ANS) EDIT(AS_OF_DATE) (X(1), P'99AAAAA99') ;
NREAD = 4;

```



```

CALL FDATE(AS_OF_DATE,CAL_DATE,0);
CAL_DATE = CAL_DATE-1;
READ#(4): PUT EDIT('ENTER THE NUMBER OF TASKS PLUS EVENTS')
(COL(1),A);
PUT EDIT(' / ____ / ') (COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(NO_DVP_STPS) (COL(2),F(5,0));
NREAD = 5;
READ#(5): PUT EDIT('ENTER THE NUMBER OF REMARKS') (COL(1),A);
PUT EDIT(' / ____ / ') (COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(NO_REMARKS) (COL(2),F(5,0));
/* ***** END SET OF REPORT HEADING ***** */
/* ***** PUT HEADING IN FILE ***** */
PUT FILE(SCRATCH) EDIT(T_SEND,CAL_DATE )
(COL(1),A,F(5,2));
PUT FILE(SCRATCH) EDIT(MAJOR_SYSTEM1,MAJOR_SYSTEM2)
(COL(1),A,A);
PUT FILE(SCRATCH) EDIT(TITLE1,TITLE2,AS_OF_DATE)
(COL(1),A,A,A);
PUT FILE(SCRATCH) EDIT(NO_DVP_STPS,NO_REMARKS)
(COL(1),(2)F(5,0));
/* ***** END HEADING INPUT TO FILE ***** */
/* ***** START ENTERING STEP DATA ***** */
XX2:NULLD = 0;
LOOP1: DO II = 1 TO NO_DVP_STPS;
NNN = 1;
CALL NEWPAGE;
CALL TSEND;
NREAD = 6;
READ#(6): PUT EDIT ('ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS')
(COL(1),A);
IF PROMPT = 'Y' I PROMPT1 = 'Y' THEN
PUT EDIT(' TASK OR EVENT ', ' FOLLOWED BY NUMBER')
(COL(2),A);
PUT SKIP(2);
GET EDIT(ANS) (COL(1), A(80)) ;
IF SUBSTR(ANS,1,1) = 'N' THEN DO;
NULLD = NULLD + 1;
GO TO NULLDVL;
END;
IF SUBSTR(ANS,1,1) = 'E' THEN NTYPE = 1;
ELSE NTYPE = 0;
NUMB = VERIFY(ANS,LETTER);
IF NUMB > 0 THEN GET STRING(ANS) EDIT(STOPI) (X(NUMB-1),F(1));

```

```

ELSE STOPI = 1;
IF NTYPE = 0 THEN DO;
IF STOPI > 3 THEN DO;
STOPI = 3;
PUT EDIT('MAX NUMBER OF SUB TASKS IS 3      ')
(COL(5),A);
END;
NREAD = 7;
READ#(7): PUT EDIT('/ ID# /          TASK          / STR.DT.
, '/ STP.DT / COMPT /') (COL(1), A, A);
PUT SKIP(2);
PUT EDIT('/* ____ */_____/ ____ */',
' ____ */_____/') (COL(1),A,A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1),A(80));
GET STRING(ANS) EDIT(ID_NO,STP_TITLE,START_DATE(NNN),STOP_DATE(NNN)
COMPLET(NNN)) (X(2), F(5,0),X(1),A(28),X(1),A(9),X(1),A(9),
X(4),F(5,2));
CALL FDATE(START_DATE(NNN),START_DATE_CODE(NNN),2);
START_DATE_CODE(NNN) = START_DATE_CODE(NNN)-1;
CALL FDATE(STOP_DATE(NNN),STOP_DATE_CODE(NNN),3);
STOP_DATE_CODE(NNN) = STOP_DATE_CODE(NNN) -1;
/* ***** */
/* ENTER MULT ELEMENTS */
/* ***** START OF INPUT OF A TASK ***** */
DO I = 2 TO STOPI;
NNN = NNN + 1;
NREAD = 8;
READ#(8): PUT EDIT('/ STR.DT. / STP.DT. / COMPT /')
(COL(1), A);
PUT EDIT('/* ____ */_____/ ____ */_____/') (COL(1), A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1),A(80));
GET STRING(ANS) EDIT(START_DATE(NNN),STOP_DATE(NNN),COMPLET(NNN))
(X(1),A(9),X(1),A(9),X(4),F(5,2)) ;
CALL FDATE(START_DATE(NNN),START_DATE_CODE(NNN),2);
START_DATE_CODE(NNN) = START_DATE_CODE(NNN)-1;
CALL FDATE(STOP_DATE(NNN),STOP_DATE_CODE(NNN),3);
STOP_DATE_CODE(NNN) = STOP_DATE_CODE(NNN) -1;
/* ***** */
END;
IF ID_NO > 0 & ID_NO < MAXNO * 10 THEN DO;
ID_NO = ID_NO/10;
TP = TIP(ID_NO);
STP_TITLE = STDTITLE;

```



```

(COL(1),F(2,0),A,(NNN)P'999V99');
END;
END LOOP1;
NULLDVL;
/* ***** END ENTERING ELEMENT DATA ***** */
/* ***** START ENTERING REMARKS ***** */
CALL NEWPAG;
CALL TSEND;
LOOP2: DO II = 1 TO NO_REMARKS;
PUT EDIT('ENTER REMARKS',II)
(COL(1),A,F(5,0));
PUT EDIT('/*_____',
'_____',
'_____*/')(COL(1),A,A,A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1),A(80));
IF SUBSTR(ANS,1,1) = 'N' THEN GO TO XXA;
GET STRING(ANS) EDIT(REMARK) (X(2), A(78));
/* ***** PUT REMARKS IN FILE ***** */
PUT FILE(SCRATCH) EDIT(REMARK) (COL(1),A(78));
/* ***** ***** */
END LOOP2;
XXA:;
/* ***** END ENTERING REMARKS ***** */
/* */
CLOSE FILE(SCRATCH) ;
XX3:;
/* ***** ***** */
DO I = 1 TO MAXNO;
TP = TIP(I);
FREE STDLE;
END;
RETURN;
/* ***** ***** */
END NEW;

```



```

NEW1: PROC;
DCL INITT ENTRY(FIXED BIN(31));
DCL NUMBER CHAR(9) INIT('123456789');
DCL TOUTPT ENTRY(FIXED BIN(31));
DCL NO_SYS FIXED DEC(5,0);
DCL PROMPT CHAR(1) EXTERNAL;
DCL PROMPT1 CHAR(1);
DCL NLABEL FIXED DEC(5,0), CTYPE CHAR(1);
DCL NCODE CHAR(1);
DCL LAB0(0:4) LABEL;
DCL SCRATCH FILE ENV(F(80,80));
NO_SYS = 1;
LABX: PUT EDIT('WILL YOU ENTER THE DATA FROM THE ',
'KEYBOARD OR IS IT PRESENTLY ON A FILE ?')
(COL(2),A,A);
PROMPT1 = 'N';
IF PROMPT = 'Y' | PROMPT1 = 'Y' THEN DO;
PUT EDIT('ENTER','K - KEYBOARD','F - FILE','N - NONE')
(COL(3),COL(5),A,COL(5),A,COL(5),A);
END;
PUT SKIP(2);
GET EDIT(CTYPE) (COL(1),A(1));
I = INDEX('NKF',CTYPE);
GO TO LAB0(I);
LAB0(0): /* CODE NOT FOUND */
PUT EDIT(' RE-ENTER CODE ')
(COL(3),A);
PUT SKIP(2);
PROMPT1 = 'Y';
GO TO LABX;
LAB0(1): /* NONE */
PUT EDIT(' NO DATA TO INPUT') (COL(1),A);
PUT SKIP(2);
/* ***** */
RETURN;
/* ***** */
LAB0(2): /* ENTER FROM KEYBOARD */
CLOSE FILE(SCRATCH);
NO_SYS = 1;
CALL NEW(NO_SYS);
NPRT = 0;
LAB0(3): /* ENTER FROM FILE */;
IF CTYPE = 'F' THEN DO;
PUT EDIT('IS YOUR DATA IN A REPORT FORM ?')
(COL(1),A); PUT SKIP(2);
GET EDIT(CTYPE) (COL(1),A(1));
IF CTYPE = 'Y' THEN NPRT = 20;

```

```

ELSE NPRT = 0;
END;
BEGIN;
DECLARE LINE CHAR(114);
DECLARE 1 CARD BASED(P),
  2 DELETE_FLAG BIT(8),
  2 ASE_CODE CHAR(7),
  2 APLI_CODE CHAR(11),
  2 MAJOR_SYSTEM1 CHAR(12),
  2 MAJOR_SYSTEM2 CHAR(20),
  2 TITLE1 CHAR(20),
  2 TITLE2 CHAR(24),
  2 AS_OF_DATE CHAR(10),
  2 NO_DVL_STPS FIXED DEC(5,0),
  2 NO_REMARKS FIXED DEC(5,0),
  2 FILE_END FIXED DEC(5,0);
DECLARE 1 DEVEL_STEP BASED(Q),
  2 DELETE_FLAG BIT(8),
  2 INFO,
  3 ASE_CODE CHAR(7),
  3 APLI_CODE CHAR(11),
  3 TYPE_NUMBER FIXED DEC(2),
  3 TITLE CHAR(28),
  3 STEP_DATES(9) FIXED DEC(5,2),
  3 COST(4) FIXED DEC(5,2),
  3 REST CHAR(26);
DECLARE 1 REMARKS BASED(R),
  2 DELETE_FLAG BIT(8),
  2 ASE_CODE CHAR(7),
  2 APLI_CODE CHAR(11),
  2 REMARK CHAR(80),
  2 REST CHAR(15);
DCL ANS CHAR(40) VAR;
DCL PP POINTER;
DCL (I0,I01) FIXED DEC(5,0);
DCL (X2,X3,X4) PIC'9999';
DCL X1 PIC'999';
DCL DUMMY CHAR(80) VAR;
DECLARE DIREC FILE RECORD KEYED ENV(INDEXED F(1140,114));
P = ADDR(LINE);
Q = P;
R = P;
/* ***** */
/*      OCFILE(1) => SEQUENTIAL UPDATE      */
/*      OCFILE(2) => DIRECT UPDATE           */
/*      OCFILE(3) => SEQUENTIAL OUTPUT       */
/* ***** */

```

```

ON ENDFILE(SCRATCH) BEGIN;
REVERT ENDFILE(SCRATCH);
GO TO FINISH;
END;
ON CONVERSION BEGIN;
REMARKS.REST = 'RRRRRRRRRRRRRRRRRR';
NOREMARKS = NOREMARKS - 1;
GET STRING(ANS) EDIT(REMARK) (A(80));
NOVLSTPS = 11;
REMARKS.ASE_CODE = ASECODE;
WRITE FILE(DIREC) FROM(REMARKS) KEYFROM(ASECODE);
GO TO L3;
END;
ON ERROR BEGIN;
PUT EDIT('YOU HAVE SOMETHING WRONG WITH YOUR MAIN FILE.',
'DO NOT WRITE REPORTS OR ENTER A NEW SYSTEM ',
'UNTIL YOU HAVE CONSULTED THE "SCHEDULE" ',
'PROGRAMMER.',
'YOU MAY CONTINUE TO EDIT DATA IF YOU WISH.')
```

```

(COL(5),A,COL(5),A,A,A);
PUT SKIP(2);
GO TO FINISH;
END;
CLOSE FILE(SCRATCH);
IO = 3;
OPEN FILE(SCRATCH) INPUT;
FILE_END = 6;
TYPE_NUMBER = 1;
CARD.DELETE_FLAG = (8)'0'B;
DCL APLICODE CHAR(11), ASECODE CHAR(7) VAR;
LABX1: PUT EDIT(' DO YOU WANT THIS DATA ADDED TO THE MAIN FILE?')
(COL(1),A);
PUT SKIP(2);
GET EDIT(ANS) (COL(1),A(80));
IF ANS = 'Y' | ANS = 'YES' THEN DO;
ON KEY(DIREC) BEGIN;
PUT EDIT(' FILE CODE FOR THE SYSTEM IS ',ASECODE)
(COL(3),A,A);
PUT SKIP(2);
GO TO LABX2;
END;
/*      *****      */
IO = 1;
X1 = 0;
CALL OCFILE(IO);
DO WHILE('1'B);
X1 = X1 + 1;

```



```

ASECODE = CHAR(X1)11'0000';
READ FILE(DIREC) KEY(ASECODE) SET(PP);
END;
LABX2: PUT EDIT('ENTER APLI CODE') (COL(1),A);
PUT SKIP(2);
NCODE = SUBSTR(ASECODE,1,1);
GET EDIT(APLICODE) (COL(1),A(11));
IO = 3;
CALL OCFILE(IO);
CARD.APLI_CODE = APLICODE;
GET FILE(SCRATCH) EDIT(DUMMY) (COL(1),A(80));
GET FILE(SCRATCH) EDIT(CARD.MAJOR_SYSTEM1,CARD.MAJOR_SYSTEM2)
(COL(1),(NPRT)X(1),A(12),A(20));
GET FILE(SCRATCH) EDIT(CARD.TITLE1,CARD.TITLE2,CARD.AS_OF_DATE)
(COL(1),A(20),A(24),A(10));
X4 = 0;
RPT:GET FILE(SCRATCH) EDIT(CARD.NO_DVL_STPS,CARD.NO_REMARKS)
(COL(1),(2)F(5,0));
NOREMARKS = CARD.NO_REMARKS;
NODVLSTPS = CARD.NO_DVL_STPS;
IF CTYPE = 'Y' & X4 = 0 THEN GO TO L2;
CARD.NO_DVL_STPS=0.0;
CARD.NO_REMARKS=0.0;
X2 = 0;
ASECODE = SUBSTR(ASECODE,1,3)11CHAR(X2);
CARD.ASE_CODE = ASECODE;
WRITE FILE(DIREC) FROM(CARD) KEYFROM(ASECODE);
L2: DO I1 = 1 TO NODVLSTPS;
X3 = I1*10 + X4;
ASECODE = SUBSTR(ASECODE,1,3)11CHAR(X3);
DEVEL_STEP.ASE_CODE = ASECODE;
GET FILE(SCRATCH) EDIT(ANS) (COL(1),A(80));
DO III = 1 TO 4;
COST(III) = 0.0;
END;
GET STRING(ANS) EDIT(NNN)(F(2,0));
TYPE_NUMBER = NNN;
MMM = ABS(NNN);
DO III = MMM TO 9;
STEP_DATES(III) = 0.0;
END;
GET STRING(ANS) EDIT(TITLE) (X(2),A(28));
IF NNN < 1 THEN MMM = 3*MMM;
GET STRING(ANS) EDIT((STEP_DATES(III) DO III = 1 TO MMM)
(X(30),(MMM)(F(5,2))));
DEVEL_STEP.REST = 'SSSSSSSSSSSSSSSSSSSSSSSSSSSS';
WRITE FILE(DIREC) FROM(DEVEL_STEP) KEYFROM(ASECODE);

```



```

END L2;
REMARKS.REST = 'RRRRRRRRRRRRRRRR';
X4 = X3;
L3: DO I2 = 1 TO NOREMARKS;
X4 = (I2 + NODVLSTPS)*10;
ASECODE = SUBSTR(ASECODE,1,3) || CHAR(X4);
REMARKS.ASE_CODE = ASECODE;
GET FILE(SCRATCH) EDIT(REMARK) (COL(1),A(80));
WRITE FILE(DIREC) FROM(REMARKS) KEYFROM(ASECODE);
END L3;
IF CTYPE = 'Y' THEN GO TO RPT;
END;
FINISH: CLOSE FILE(SCRATCH);
CLOSE FILE(DIREC);
END;
/*      *****      */
RETURN;
/*      *****      */
END NEW1;

```

```

RDATE: PROC(DAY_MONTH_YEAR,DATE_CODE);
DECLARE DAY_MONTH_YEAR CHARACTER(9);
DECLARE (MONTH) CHARACTER(3);
DCL (YEAR,MONTH_CODE) FIXED DEC(5,2);
DECLARE DAY_FACTOR(12) FIXED DEC(5,2)
INIT(31.0,28.0,31.0,30.0,
31.0,30.0,31.0,31.0,
30.0,31.0,30.0,31.0);
DECLARE DATE(12) CHARACTER(3)
INIT('JAN','FEB','MAR','APR',
'MAY','JUN','JUL','AUG',
'SEP','OCT','NOV','DEC');
DECLARE START_YEAR FIXED DEC(5,2) INIT(76.0);
DATE_CODE FIXED DEC(5,2),IDAY FIXED DEC(5,2);
DECLARE (C1,C3) CHAR(2);
DECLARE C2 CHAR(3);
DECLARE (IDAY1,IYR) PIC'99';
DECLARE IDATE FIXED DEC(3);
DECLARE OFFSET FIXED DEC(2) INIT(0);
/* ***** START FINDING MONTH ***** */
IDATE = TRUNC(DATE_CODE);
IMON = MOD(IDATE,12);
IF IMON = 0 THEN DO;
IMON = 12;
OFFSET = 1;
END;
C2 = DATE(IMON);
/* ***** END FINDING MONTH CODE ***** */
IYR = TRUNC(DATE_CODE/12) + START_YEAR - OFFSET;
IDAY = DATE_CODE - TRUNC(DATE_CODE);
IDAY1 = IDAY*DAY_FACTOR(IMON);
IF IDAY1 = 0 THEN DO;
IF IMON = 1 THEN DO;
IMON = 13;
IYR = IYR - 1;
END;
C2 = DATE(IMON-1);
IDAY1 = DAY_FACTOR(IMON-1);
END;
C1 = CHAR(IDAY1);
C3 = CHAR(IYR);
/* ***** CALCULATE DATE ***** */
DAY_MONTH_YEAR = C1||' '||C2||' '||C3;
/* ***** */
END RDATE;

```

```

REPORTS: PROC(IX,LTYPE);
DCL (NSARY,NRARY) FIXED DEC(5) EXTERNAL;
DCL PROMPT CHAR(1) EXTERNAL;
DCL PQ CHAR(1);
DCL (JENNY1,JENNY2,JENNY3,JENNY4,JENNY5) CHAR(1) EXTERNAL;
DCL (LLIMIT,ULIMIT,STYR,NOYR) FIXED DEC(5,2) EXTERNAL;
DCL SCRATCH FILE ENV(F(80,80));
DCL LAB0(0:1) LABEL;
DCL PROMPT1 CHAR(1);
DCL LINE CHAR(114);
DCL PM POINTER;
DCL PS(*) POINTER CTL;
DCL PR(*) POINTER CTL;
DCL LLINE CHAR(114) BASED(PP);
DCL INITT ENTRY(FIXED BIN(31));
DCL TOUTPT ENTRY(FIXED BIN(31));
DECLARE 1 CARD BASED(P),
    2 DELETE_FLAG BIT(8),
    2 ASE_CODE CHAR(7),
    2 APLI_CODE CHAR(11),
    2 MAJOR_SYSTEM1 CHAR(12),
    2 MAJOR_SYSTEM2 CHAR(20),
    2 TITLE1 CHAR(24),
    2 TITLE2 CHAR(20),
    2 AS_OF_DATE CHAR(10),
    2 ST_YR FIXED DEC(5,0),
    2 NO_YR FIXED DEC(5,0),
    2 FILE_END FIXED DEC(5,0);
DECLARE 1 DEVEL_STEP BASED(Q),
    2 DELETE_FLAG BIT(8),
    2 INFO,
    3 ASE_CODE CHAR(7),
    3 APLI_CODE CHAR(11),
    3 TYPE_NUMBER FIXED DEC(2),
    3 TITLE CHAR(28),
    3 STEP_DATES(9) FIXED DEC(5,2),
    3 COST(4) FIXED DEC(5,2),
    3 REST CHAR(26);
DECLARE 1 REMARKS BASED(R),
    2 DELETE_FLAG BIT(8),
    2 ASE_CODE CHAR(7),
    2 APLI_CODE CHAR(11),
    2 REMARK CHAR(80),
    2 REST CHAR(15);
DCL AGENCY CHAR(20);
DCL REF_DT DEC(5,0) INIT(76);
DCL T_SEND CHAR(40);

```

```

DCL (IO,IO1) FIXED DEC(5,0);
DCL CAL_DATE FIXED DEC(5,2);
/* ***** */
DCL DIREC FILE RECORD KEYED ENV(INDEXED F(1140,114));
DCL ASECOD CHAR(7) VAR;
DCL ANS CHAR(80) VAR;
DCL (UDTYPE,CTYPE,LTYPE,NCROSS) CHAR(1);
DCL CTYPE4 CHAR(2);
DCL L1 PIC'999',L2 PIC'9999';
DCL FILL CHAR(5);
DCL (NLABEL,II,IX,IY) FIXED DEC(5,0);
DCL START_DATE_CODE(5) FIXED DEC(5,2);
DCL STOP_DATE_CODE(5) FIXED DEC(5,2);
DCL COMPLET(5) FIXED DEC(5,2);
DCL ASECOD1 CHAR(7);
DCL APLICOD CHAR(11) VAR;
DCL APLICOD2 CHAR(12) VAR;
DCL STITLE CHAR(1);
DCL NSANS CHAR(1);
DCL TARRAY(11) CHAR(40)
INIT('          BUDGET
IX = 0;
NJFN = 30;
ASECOD1 = 'XXXXXXXX';
ALLOCATE PS(NSARY),PR(NRARY);
BEGIN;
DCL DSTR CHAR(6);
DCL NDYPM(12) FIXED DEC(5,2) INIT(31.,28.,31.,30.,31.,30.,
31.,31.,30.,31.,30.,31);
DSTR = DATE;
CAL_DATE = (DEC(SUBSTR(DSTR,1,2))-76)*12
+ DEC(SUBSTR(DSTR,3,2))-99
+ DEC(SUBSTR(DSTR,5,2),5,2)/NDYPM(DEC(SUBSTR(DSTR,3,2)));
END;
NS = 0;
/*      UPDATING FILE      */
/*      *****      */
/* OCFILE(1) = SEQUENTIAL */
/* OCFILE(2) = DIRECT      */
/*      *****      */
IO = 1;
OPEN FILE(SCRATCH) OUTPUT;
OPEN FILE(DIREC) SEQUENTIAL UPDATE;
L1 = 0;
PUT EDIT('HOW MANY SYSTEMS DO YOU WANT TO SEE?')
(COL(1),A);
PUT EDIT('ENTER') (COL(3),A);

```



```

IF PROMPT = 'Y' THEN PUT EDIT('A - ALL',
'S - SUBSET OF ALL',
'N - NONE')
(COL(5),A);
PUT SKIP(2);
GET EDIT(ANS) (COL(1),A(80));
CALL ER_PAGE;
IF ANS = 'A' & JENNY2 = 'C' THEN DO;
PUT EDIT('DO YOU WANT ALL SYSTEMS ON ONE REPORT ?')
(COL(1),A); PUT SKIP(2);
GET EDIT(NCROSS) (COL(1),A(1));
END;
IF ANS = 'N' / ANS = 'NONE' THEN GO TO LAB2;
PUT EDIT('ENTER OFFICE PREPARING REPORTS')
(COL(1),A);
PUT SKIP(2);
GET EDIT(AGENCY) (COL(1),A(20));
IF JENNY4 = 'M' THEN DO;
PUT EDIT('DO YOU WANT THE MAXIMUM LINES PER PAGE TO BE 30')
(COL(1),A); PUT SKIP(2);
GET EDIT(NSANS) (COL(1),A(1));
IF NSANS = 'Y' THEN CALL CHG_MAX;
END;
CHG_MAX:PROC;
ON CONVERSION BEGIN;
PUT EDIT('ENTER AGAIN PLEASE.',
'BE SURE IT IS A NUMBER.')
```

```

(COL(1),A); PUT SKIP(2);
REVERT CONVERSION;
GO TO JBA34;
END;
JBA34:PUT EDIT('ENTER NEW MAXIMUM NUMBER OF LINES PER PAGES')
(COL(1),A); PUT SKIP(2);
GET EDIT(NJEN) (COL(1),F(2,0));
IF NJEN > 30 THEN DO; PUT EDIT(' MAXIMUM LINES PER PAGE IS 30')
(COL(1),A); PUT SKIP(2);
NJEN = 30;
END;
RETURN;
END CHG_MAX;
IF ANS = 'A' THEN CALL JA;
IF APLICODE = 'N' THEN GO TO LAB2;
JA:PROC;
PUT EDIT(' ENTER APPLICATION OF INTEREST')
(COL(1),A);
IF PROMPT = 'Y' / PROMPT1 = 'Y' THEN DO;
IF JENNY1 = 'P' THEN
```

```

PUT EDIT('A - ALL','B - BUDGET','E - PROJECT ENGINEER',
'H - PROJECT MANAGER','O - 104 REPORT',
'P - PROCUREMENT','Q - PRODUCT ASSURANCE',
'R - RECAP','S - PROGRAM SUMMARY','T - TEST','N - NONE')
(COL(5),A);
IF JENNY1 = 'R' THEN
PUT EDIT('A - ALL','C - COMPTROLLER','B - PLANS & ANALYSIS',
'P - PROCUREMENT','E - DEVELOPMENT & ENGINEERING',
'F - ASTIO','T - TRADOC','S - TECOM',
'Q - PRODUCT ASSURANCE','N - NONE')
(COL(5),A);
END;
PUT SKIP(2);
GET EDIT(APLICODE2) (COL(1),A(12));
LSTR = INDEX(APLICODE2,' ') - 1;
APLICODE = SUBSTR(APLICODE2,1,LSTR);
IF APLICODE = 'N' THEN GO TO LABJBA;
IF INDEX(APLICODE,'A') /= 0 THEN DO;
APLICODE = 'A';
GO TO LAB3;
END;
IF LENGTH(APLICODE) = 1 & JENNY1 /= 'P' THEN DO;
PUT EDIT(' DO YOU WANT A STANDARD TITLE?')
(COL(1),A);
IF PROMPT = 'Y' | PROMPT1 = 'Y' THEN
PUT EDIT('Y - YES','N - NO')
(COL(5),A);
PUT SKIP(2);
GET EDIT(STITLE) (COL(1),A(1));
IF STITLE = 'Y' THEN DO;
ITIT = INDEX('BEHOPQRST',APLICODE);
IF ITIT = 0 THEN GO TO LAB3;
T_SEND = TARKAY(ITIT);
GO TO LABJBA;
END;
END;
LAB3: PUT EDIT(' ENTER REPORT TITLE') (COL(1),A);
PUT SKIP(2);
GET EDIT(T_SEND) (COL(1),A(40));
LABJBA:;
CALL ER_PAGE;
END JA;
ON ENDFILE (DIREC) BEGIN;
REVERT ENDFILE(SCRATCH);
IF NS > 0 THEN GO TO XX;
ELSE;
IF IX = 1 THEN GO TO LAB2;

```

```

ELSE PUT EDIT('NO DATA PREPARED FOR REPORTS') (COL(2),A);
PUT SKIP(2);
GO TO LAB2;
END;
ON KEY(DIREC) BEGIN;
IF ONCODE = 51 THEN DO;
PUT EDIT(ASECODE,'RECORD NOT FOUND')
(COL(1),A,A);
IF ANS = 'A' | ANS = 'ALL' THEN GO TO LAB2;
PUT SKIP(2);
GO TO LAB0(0);
END;
END;
LAB0(0): IF ANS = 'A' | ANS = 'ALL' THEN DO;
L1 = L1 + 1;
ASECODE = CHAR(L1) || '0000';
GO TO LAB4;
END;
PUT EDIT('ENTER SYSTEM CODE ') (COL(1),A);
IF PROMPT = 'Y' | PROMPT1 = 'Y' THEN
PUT EDIT('XXX - SYSTEM CODE',
(COL(5),A);
ELSE PUT SKIP(2);
PUT SKIP(2);
GET EDIT(ASECODE) (COL(1),A(7));
IF ASECODE1 = ASECODE | ASECODE1 = 'XXXXXXX' THEN CALL JA;
IF ASECODE = 'N' | APLICODE = 'N' THEN GO TO LAB2;
LAB4: ASECODE1 = ASECODE;
ASECODE = SUBSTR(ASECODE,1,3) || '0000';
ALLOCATE CARD SET(P);
READ FILE(DIREC) INTO(CARD) KEY(ASECODE);
NS = 0;
NR = 0;
DO WHILE('1'B);
NCRS:ALLOCATE LLINE SET(PP);
READ FILE(DIREC) INTO(LLINE) KEYTO(ASECODE);
IF SUBSTR(ASECODE,4,4) = '0000' THEN DO;
IF NCRSS ~= 'Y' THEN GO TO XX;
ELSE DO;
FREE LLINE;
GO TO NCRS;
END;
END;
ELSE;
IF APLICODE = 'A' |
(INDEX(SUBSTR(LLINE,9,11),APLICODE) ~= 0 & JENNY3 = 'N') |
(INDEX(SUBSTR(LLINE,9,LSTR),APLICODE) ~= 0 & JENNY3 = 'P')

```

```

THEN LOOPAPL: DO;
IF SUBSTR(LLINE,113,1) = 'S' THEN DO;
IF LTYPE = 'L' THEN DO;
Q = PP;
IF TYPE_NUMBER > 0 THEN DO JBA3 = 1 TO TYPE_NUMBER;
IF STEP_DATES(JBA3) >= LLIMIT &
STEP_DATES(JBA3) <= ULIMIT THEN GO TO NOX;
END;
ELSE DO;
NTYNU = - TYPE_NUMBER;
DO JBA3 = 1 TO 7 BY 3 WHILE(NTYNU*3 > JBA3);
IF (STEP_DATES(JBA3)<=LLIMIT &
STEP_DATES(JBA3+1)>=ULIMIT) |
(STEP_DATES(JBA3)>=LLIMIT &
STEP_DATES(JBA3)<=ULIMIT) |
(STEP_DATES(JBA3+1)>=LLIMIT &
STEP_DATES(JBA3+1)<=ULIMIT)
THEN GO TO NOX;
END;
END;
DO;
FREE LLINE;
GO TO NOX1;
END;
END;
NOX::
NS = NS + 1;
IF NS > NSARY THEN
CALL TBIG;
PS(NS) = PP;
END;
ELSE DO;
NR = NR + 1;
IF NR > NSARY THEN
CALL TBIG;
PR(NR) = PP;
END;
END LOOPAPL;
ELSE FREE LLINE;
NOX1::
END;
XX: IF MOD(NS,NJEN) /= 0 THEN NSPAGES = NS/NJEN + 1;
ELSE NSPAGES = NS/NJEN;
IF MOD(NR,6) /= 0 THEN NRPAGES = NR/6 + 1;
ELSE NRPAGES = NR/6;
IF NRPAGES > NSPAGES THEN NNPAGES = NRPAGES;
ELSE NNPAGES = NSPAGES;

```



```

FREE LLINE;
/* PUT SCRATCH EVERY THING EXCEPT DVL STP NO */
/* ***** */
IF ULIMIT > 999.00 THEN DO;
STYR = ST_YR;
NOYR = NO_YR;
END;
PUT FILE(SCRATCH) EDIT(T_SEND,CAL_DATE,REF_DT,STYR,NOYR)
(COL(1),A,P'999V99',(3)F(5,0));
PUT FILE(SCRATCH) EDIT(AGENCY,MAJOR_SYSTEM1,MAJOR_SYSTEM2)
(COL(1),A,A,A);
PUT FILE(SCRATCH) EDIT(TITLE1,TITLE2,AS_OF_DATE)
(COL(1),A,A,A);
IX = 1;
LS2 = 0;
LR2 = 0;
IF NNPAGES = 0 THEN PUT FILE(SCRATCH)
EDIT(LS2,LS2,LS2+1,NNPAGES+1)
(COL(1),(4)F(5,0));
LOOPA: DO LA = 1 TO NNPAGES;
LS3 = LS2;
LR3 = LR2;
LS1 = (LA-1) * NJEN + 1;
LS2 = LS1 + NJEN - 1;
IF LS2 > NS THEN LS2 = NS;
ELSE;
LR1 = (LA-1) * 6 + 1;
LR2 = LR1 + 5;
IF LR2 > NR THEN LR2 = NR;
ELSE;
IF LA = 1 THEN DO;
LS3 = LS2;
LR3 = LR2;
GO TO SS;
END;
LS3 = LS2 - LS3;
LR3 = LR2 - LR3;
SS:;
PUT FILE(SCRATCH) EDIT(LS3,LR3,LA,NNPAGES)
(COL(1),(4)F(5,0));
LOOPS: DO LS = LS1 TO LS2;
Q = PS(LS);
/* PUT DVL STP IN FILE */
LAB0(1): /* DEVEL STEP */
NNNN = TYPE_NUMBER;
NNN = ABS(NNNN);
IF NNNN < 0 THEN NN = NNN*3;

```

```

ELSE NN = NNN;
PUT FILE(SCRATCH) EDIT(NNNN,TITLE,
(STEP_DATES(III) DO III=1 TO NN))
(COL(1),F(2,0),A,(NN)P'999V99');
FREE DEVEL_STEP;
END LOOPS;
LOOPR: DO LR = LR1 TO LR2;
R = PR(LR);
/* PUT REMARK IN FILE */
PUT FILE(SCRATCH) EDIT(REMARK) (COL(1),A(72));
FREE REMARKS;
END LOOPR;
END LOOPA;
PUT EDIT('SYSTEM ',SUBSTR(ASECODE1,1,3),' IS READY.')
(COL(1),A,A,A);
PUT SKIP(2);
IX = 1;
FREE CARD;
IF NCROSS .NE. 'Y' THEN GO TO LAB0(0);
ELSE;
LAB2:;
CLOSE FILE(SCRATCH);
CLOSE FILE(DIRECT);
/*      *****      */
RETURN;
/*      *****      */
ER_PAGE:PROC;
CALL NEWPAGE;
CALL TSEND;
RETURN;
END ER_PAGE;
TBIG:PROC;
PUT EDIT('THE REPORT IS LIMITED TO ',NSARY,
' DEVELOPMENT STEPS AND ',NARY,
' REMARKS.', ' YOU HAVE WRITTEN',NS,
' DEVELOPMENT STEPS AND ',NR,
' REMARKS AND ARE NOT FINISHED.')
(COL(1),(2)(A,F(5,0)));
PUT EDIT('DO YOU WANT TO SEE THE REPORTS ',
'AS WRITTEN OR TRY AGAIN?') (COL(1),A,A);
PUT SKIP(2);
IF PROMPT = 'Y' | PROMPT1 = 'Y' THEN
PUT EDIT('ENTER', 'P - PARTIAL REPORT',
'Q - QUIT')
(COL(1),A,COL(5),A);
GET EDIT(PQ) (COL(1),A(1));
FREE LLINE;

```

IF PQ = 'P' THEN GO TO XX;
IX = 0;
GO TO LAB2;
END TRIG;
END REPORTS;

```

SSLIP: PROC(ASECODE);
DCL LETTER1 CHAR(26) INIT('ABCDEFGHIJKLMNOPQRSTUVWXYZ');
DCL SDATE FIXED DEC(5,2);
DCL FDATE ENTRY(CHAR(9),FIXED DEC(5,2),FIXED BIN(15));
DCL XDATE CHAR(9);
DCL PROMPT CHAR(1) EXTERNAL;
DCL PROMPT1 CHAR(1);
DCL (JENNY1,JENNY2,JENNY3,JENNY4,JENNY5) CHAR(1) EXTERNAL;
DCL LLINE CHAR(114);
DECLARE 1 DEVEL_STEP BASED(Q),
      2 DELETE_FLAG BIT(8),
      2 INFO,
      3 ASE_CODE CHAR(7),
      3 APLI_CODE CHAR(11),
      3 TYPE_NUMBER FIXED DEC(2),
      3 TITLE CHAR(28),
      3 STEP_DATES(9) FIXED DEC(5,2),
      3 COST(4) FIXED DEC(5,2),
      3 REST CHAR(26);
DCL (I0,I01) FIXED DEC(5,0);
/* ***** */
DECLARE DIREC FILE RECORD KEYED ENV(INDEXED F(1140,114));
DCL ASECODE CHAR(7);
DCL ANS CHAR(80) VAR;
DCL APLICODE CHAR(11) VAR;
DCL MONTHS FIXED DEC(6,2),(ANS1,ANS2,ANS3) CHAR(1);
DCL FILL CHAR(5);
DCL (NLABEL,II,IX,IY) FIXED BIN;
DCL TOUTPT ENTRY(FIXED BIN(31));
DCL INITT ENTRY(FIXED BIN(31));
ON ENDFILE(DIREC) BEGIN;
PUT EDIT('END OF MAIN FILE') (COL(1),A);
GO TO XX0;
END;
PUT EDIT('HOW MANY RECORDS DO YOU WANT TO SLIP ?') (COL(1),A);
IF PROMPT='Y' I PROMPT1='Y' THEN
PUT EDIT('A - ALL DATES(START&STOP) THAT OCCUR AFTER DATE X',
'S - ONLY STOP DATES THAT OCCUR AFTER DATE X',
'N - NONE')
(COL(5),A);
PUT SKIP(2);
GET EDIT(ANS1) (COL(1),A(1));
IF ANS1 = 'S' THEN DO;
K1=2;K2=5;K4=8;K5,K7,K8=10;
END;
ELSE DO;
K1=1;K2=2;K4=4;K5=5;K7=7;K8=8;

```



```

END;
IF ANS1 = 'N' THEN GO TO XX2;
PUT EDIT('ENTER SLIP REFERENCE DATE X') (COL(1),A);
PUT EDIT('/_*__*/') (COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1),A(80));
GET STRING(ANS) EDIT(XDATE) (X(1),P'99AAAAA99');
CALL FDATE(XDATE,SDATE,0);
SDATE = SDATE - 1;
PUT EDIT('SHOULD THE SLIP DEPEND ON APPLICATION CODE ?') (COL(1),A);
IF PROMPT = 'Y' | PROMPT1 = 'Y' THEN
PUT EDIT('ENTER','Y - YES','N - NO') (COL(1),A,(2)(COL(5),A));
PUT SKIP(2);
GET EDIT(ANS2) (COL(1),A(1));
IF ANS2 = 'N' THEN GO TO XX1;
PUT EDIT('ENTER APPLICATION CODE FOR STEPS TO BE SLIPPED') (COL(2),A);
PUT EDIT('/_*/') (COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1),A(80));
IF SUBSTR(ANS,1,1) = 'N' THEN GO TO XX2;
LSTR = INDEX(SUBSTR(ANS,2,12),' ') - 1;
APLICODE = SUBSTR(ANS,2,LSTR);
XX1: PUT EDIT('ENTER NUMBER OF MONTHS OF SLIPPAGE','/_*/')
(COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1),A(80));
GET STRING(ANS) EDIT(MONTHS) (X(1),F(6,2));
PUT EDIT(MONTHS) (COL(1),F(6,2));
PUT EDIT('DO WE AGREE ON THE NUMBER OF MONTHS ?') (COL(1),A);
PUT SKIP(2);
GET EDIT(ANS3) (COL(1),A(1));
IF ANS3 = 'N' THEN GO TO XX1;
IF ANS1 = 'A' THEN N = -1;
ELSE N = 0;
L1: DO I = 1 TO 10000;
II = 1;
READ FILE(DIREC) INTO(LLINE) KEYTO(ASECODE);
IF SUBSTR(ASECODE,4,4) = '0000' THEN GO TO XX0;
Q = ADDR(LLINE);
IF SUBSTR(LLINE,113,1) = 'S' THEN L2:DO;
IF (ANS2 = 'N') &
((INDEX(SUBSTR(LLINE,9,11),APLICODE) = 0 & JENNY3 = 'N') |
(INDEX(SUBSTR(LLINE,9,LSTR),APLICODE) = 0 & JENNY3 = 'P'))
THEN NAPLI = 1;

```

```

ELSE NAPLI = 0;
IF ANS2 = 'N' | NAPLI /= 0 THEN L3:DO;
IF TYPE_NUMBER > 0 THEN DO K= 1 TO TYPE_NUMBER;
IF STEP_DATES(K) > SDATE THEN
STEP_DATES(K) = STEP_DATES(K) + MONTHS;
END;
IF TYPE_NUMRER < 0
THEN DO K = K1,K2,K4,K5,K7,K8 WHILE (TYPE_NUMBER * 3<-K);
IF STEP_DATES(K) > SDATE & STEP_DATES(K) /= 0 THEN
STEP_DATES(K) = STEP_DATES(K) + MONTHS;
END;
END L3;
END L2;
REWRITE FILE(DIREC) FROM(LLINE);
END L1;
XX0: PUT EDIT(' SLIP COMPLETED.') (COL(1),A);
PUT SKIP(2);
XX2:RETURN;
END SSLIP;

```

```

TYREPT: PROC ;
DCL PR_BY CHAR(20);
DCL CAL_DATE CHAR(9);
DCL DUMCH CHAR(80) VAR;
DCL ALDATE(10) CHAR(9);
DCL ANS CHAR(80) VAR;
DCL MAJOR_SYSTEM CHAR(32), RPTIT CHAR(40),
TITLE1 CHAR(44), AS_OF_DATE CHAR(10),
CH CHAR(1);
DCL (STEP_DATE, CUR_DT, DUMMY) FIXED DEC(5,2),
TYPE_NUMBER FIXED DEC(2,0);
DCL (NO_DVLS, NO_RMKS, NPG, NPG_OF_NPGS, I, II, J, KKA) FIXED BIN;
DCL SCRATCH FILE ENV(F(80,80));
OPEN FILE(SCRATCH) INPUT;
ALDATE(10) = (9) ' ';
/* ***** */
ON ENDFILE(SCRATCH) GO TO FINISH;
DUMRD: PROC;
NPG = J;
GET FILE(SCRATCH) EDIT(NO_DVLS, NO_RMKS) (COL(1), (2)F(5,0));
RETURN;
END DUMRD;
NEXT: GET FILE(SCRATCH) EDIT(RPTIT, CUR_DT) (COL(1), A(40), F(5,2));
GET FILE(SCRATCH) EDIT(PR_BY, MAJOR_SYSTEM) (COL(1), A(20), A(32));
GET FILE(SCRATCH) EDIT(TITLE1, AS_OF_DATE) (COL(1), A(44), A(10));
GET FILE(SCRATCH) EDIT(NO_DVLS, NO_RMKS, NPG, NPG_OF_NPGS)
(COL(1), (4)F(5,0));
PUT SKIP(2);
PUTHEAD: PROC;
DUMMY = CUR_DT + 1.0;
CALL RDATE(CAL_DATE, DUMMY);
PUT EDIT(' ' DO II = 2 TO 72) (COL(2), 72 A(1));
PUT EDIT(' ', RPTIT, ' ') (COL(1), A(1), X(20), A, COL(73), A);
PUT EDIT(' ', (' ' DO II = 1 TO 71), ' ') (SKIP(0), COL(1), A(1),
71 A(1), A(1));
PUT EDIT(' ', 'CURRENT DATE:', CAL_DATE, 'PAGE', NPG, 'OF', NPG_OF_NPGS, '
(COL(1), A(1), A, X(2), A, COL(57), A, X(1), P'Z9', X(1), A, X(1), P'Z9', COL(73
, A(1));
PUT EDIT(' ', (' ' DO II = 1 TO 71), ' ') (SKIP(0), COL(1), A(1), 71 A(1)
A(1));
PUT EDIT(' ', 'PREPARED BY:', PR_BY, ' ') (COL(1), A, A, A, COL(73), A);
PUT EDIT(' ', 'MAJOR SYSTEM:', MAJOR_SYSTEM, ' ') (COL(1), A(1), A, X(1),
, COL(73), A(1));
PUT EDIT(' ', 'PROJECT TITLE:', TITLE1, ' ') (COL(1), A(1), A, X(1), A,
COL(73), A);
PUT EDIT(' ', 'AS OF DATE:', AS_OF_DATE, ' ') (COL(1), A(1), A, X(1), A,
COL(73), A);

```



```

PUT EDIT(' ','('_ DO II = 1 TO 71),'') (SKIP(0),COL(1),A(1),71 A(1
A(1)))
PUT EDIT(' ','TASK/EVENT','START/EVENT','STOP/EVENT','STATUS','')
(COL(1),A(1),COL(10),A,X(19),A,X(2),A,X(4),A,COL(73),A)
PUT EDIT(' ','DATE' DO II = 1 TO 2),'')
(COL(1),A(1),X(40),A,X(9),A,COL(73),A)
PUT EDIT(' ','
(' DO II = 1 TO 71),'') (COL(1),A(1)
,71 A(1), A(1))
END PUTHEAD
RPT: DO J = 1 TO NPG_OF_NPGS
IF J > 1 THEN CALL DUMRD
CALL PUTHEAD
DVLS: DO I = 1 TO NO_DVLS
GET FILE(SCRATCH) EDIT(ANS) (COL(1),A(80))
PUT EDIT(' ','I',' ',SUBSTR(ANS,3,28))
(COL(1),A(1),X(2),F(2),A,X(1),A)
GET STRING(ANS) EDIT(TYPE_NUMBER) (F(2,0))
IF TYPE_NUMBER > 0 THEN DO
LE: DO K = 1 TO TYPE_NUMBER
GET STRING(ANS) EDIT(STEP_DATE) (X(30+(K-1)*5),F(5,2))
IF STEP_DATE < 12.015 THEN DO
ALDATE(K) = ' '
GO TO YY1
END
DUMMY = STEP_DATE + 1
IF DUMMY > 0 THEN CALL RDATE(ALDATE(K),DUMMY)
YY1:
END LE
IF TYPE_NUMBER >= 2 THEN KKA = 2
ELSE KKA = 1
PUT EDIT((ALDATE(K) DO K = 1 TO KKA),'')
(COL(36),(KKA)(X(4),A),COL(73), A(1))
DO KKK = 3 TO TYPE_NUMBER BY 2
IF KKK >= TYPE_NUMBER THEN KKA = 1
ELSE KKA = 2
PUT EDIT(' ','(ALDATE(KK+KKK-1) DO KK = 1 TO KKA),'')
(COL(1),A,COL(40),(KKA)(A,X(4)),COL(73),A(1))
END
END
ELSE IC = 1
NUMBER = -2*TYPE_NUMBER + 1
IF TYPE_NUMBER < 0 THEN DO
LS: DO KK = 1,2,4,5,7,8
GET STRING(ANS) EDIT(STEP_DATE) (X(30+(KK-1)*5),F(5,2))
IC = IC + 1
IF STEP_DATE < 12.015 THEN DO

```



```

ALDATE(KK) = ' ' ;
GO TO YY2 ;
END ;
DUMMY = STEP_DATE + 1 ;
IF DUMMY > 0 THEN CALL RDATE(ALDATE(KK), DUMMY) ;
ELSE GO TO YY ;
IF NUMBER <= IC THEN GO TO YY ;
YY2 ;
END LS ;
YY ;
PUT EDIT(ALDATE(1), ALDATE(2), SUBSTR(ANS, 43, 3), ' | ')
(COL(40), A, X(4), A, X(6), A(3), COL(73), A(1)) ;
PUT EDIT(' | ', ALDATE(K), ALDATE(K+1), SUBSTR(ANS, 33+(K+1)*5, 3), ' | ')
DO K = 4 TO IC BY 3)
(COL(1), A, COL(40), A, X(4), A, X(6), A(3), COL(73), A(1)) ;
END ;
PUT SKIP(2) ;
END DVLS ;
PUT EDIT(' | ', ('_' DO II = 1 TO 71), ' | ') (COL(1), A(1), 71 A(1), A(1))
CH = 'S' ;
RMKS: DO I = 1 TO NO_RMKS ;
GET FILE(SCRATCH) EDIT(ANS) (COL(1), A(80)) ;
PUT EDIT(ANS) (COL(2), A(80)) ;
END RMKS ;
PUT SKIP(4) ;
END RPT ;
GO TO NEXT ;
FINISH ;
RETURN ;
END TYREPT ;

```

```

UPDATE: PROC;
DCL ASECODE1 CHAR(7);
DCL (JENNY1,JENNY2,JENNY3,JENNY4,JENNY5) CHAR(1) EXTERNAL;
DCL FIN CHAR(1);
DCL PROMPT CHAR(1) EXTERNAL;
DCL (LAB2(0:8),LAB3(0:4),LAB6(0:4)) LABEL;
DCL PROMPT1 CHAR(1);
DCL LINE CHAR(114);
DCL 1 HCARD BASED(S),
    2 DFLAG BIT(8),
    2 REST CHAR(113);
DECLARE 1 CARD BASED(P),
    2 DELETE_FLAG BIT(8),
    2 ASE_CODE CHAR(7),
    2 APLI_CODE CHAR(11),
    2 MAJOR_SYSTEM1 CHAR(12),
    2 MAJOR_SYSTEM2 CHAR(20),
    2 TITLE1 CHAR(20),
    2 TITLE2 CHAR(24),
    2 AS_OF_DATE CHAR(10),
    2 NO_DVL_STPS FIXED DEC(5,0),
    2 NO_REMARKS FIXED DEC(5,0),
    2 FILE_END FIXED DEC(5,0);
DECLARE 1 DEVEL_STEP BASED(Q),
    2 DELETE_FLAG BIT(8),
    2 INFO,
    3 ASE_CODE CHAR(7),
    3 APLI_CODE CHAR(11),
    3 TYPE_NUMBER FIXED DEC(2),
    3 TITLE CHAR(28),
    3 STEP_DATES(9) FIXED DEC(5,2),
    3 COST(4) FIXED DEC(5,2),
    3 REST CHAR(26);
DECLARE 1 REMARKS BASED(R),
    2 DELETE_FLAG BIT(8),
    2 ASE_CODE CHAR(7),
    2 APLI_CODE CHAR(11),
    2 REMARK CHAR(80),
    2 REST CHAR(15);
DCL (I0,I01) FIXED DEC(5,0);
/* ***** */
DECLARE DIREC FILE RECORD KEYED ENV(INDEXED F(1140,114));
DCL CHG FILE ENV(F(380,38));
DCL ACHG FILE STREAM ENV(F(1140,114));
DCL ASECODE CHAR(7);
DCL ANS CHAR(80) VAR;
DCL APLICODE CHAR(11) VAR;

```

```

DCL APLICODE2 CHAR(12) VAR;
DCL (UDTYPE,CTYPE,LTYPE) CHAR(1);
DCL CTYPE4 CHAR(2);
DCL L1 PIC'999',L2 PIC'9999';
DCL FILL CHAR(5);
DCL (NLABEL,II,IX,IY) FIXED BIN;
DCL TOUTPT ENTRY(FIXED BIN(31));
DCL INITT ENTRY(FIXED BIN(31));
DCL TEMP_FLAG BIT(8);
DCL CUR_DATE CHAR(6);
DCL 1 CHGN_RECORD,
    2 ACODE CHAR(3),
    2 CTIT1 CHAR(28),
    2 DT CHAR(6),
    2 TC CHAR(1);
IDUP = 0;
/*      *****      */
P = ADDR(LINE);
Q = P;
R = P;
S = P;
/*      *****      */
CUR_DATE = DATE;
/*      UPDATING FILE      */
/*      *****      */
/* OCFILE(1) => SEQUENTIAL */
/* OCFILE(2) => DIRECT      */
/*      *****      */
IO = 2;
CALL OCFILE(IO);
ON ENDFILE(DIREC) BEGIN;
PUT EDIT(' END OF MAIN FILE') (COL(1),A);
PUT SKIP(2);
GO TO LAB2(I);
END;
ON KEY(DIREC) BEGIN;
IF ONCODE = 51 THEN DO;
PUT EDIT(ASECODE,'RECORD NOT FOUND')
(COL(1),A,A);
PUT SKIP(2);
GO TO LAB2(I);
END;
IF ONCODE = 52 THEN DO;
PUT EDIT('DUPLICATE RECORD',ASECODE)
(COL(1),A,A);
PUT SKIP(2);
GO TO LAB2(I);

```

```

END;
END;
CALL ER_PAGE;
LAB100: PUT EDIT ('ENTER EDIT CODE') (COL(1),A);
IF PROMPT = 'Y' | PROMPT1 = 'Y' THEN
PUT EDIT('A - ADD','D - DELETE','C - CHANGE',
'S - SLIP','L - LIST',
'N - NONE')
(COL(5),A);
PUT SKIP(2);
CTYPE = 'Q';
GET EDIT(UDTYPE,FILL) (COL(1),A(1),A(5));
CALL ER_PAGE;
FIN = 'Q';
PROMPT1 = 'N';
ICOUNT = 1;
I = INDEX('ADCSRLFN',UDTYPE);
/* SET FILE */
IF (I>3) & (I<7) THEN IO1 = 1;
ELSE IO1 = 2;
IF I = 4 THEN IO1 = 4;
ELSE;
IF IO1 = IO1 THEN CALL OCFILE(IO1);
IO = IO1;
/*
GO TO LAB2(1);
LAB2(0): /* ILLEGAL EDIT CODE */
PUT EDIT('EDIT CODE NOT FOUND') (COL(1),A);
PUT SKIP(2);
PROMPT1 = 'Y';
GO TO LAB100;
LAB2(1): /* ADD A RECORD */
TC = 'A';
CARD.DELETE_FLAG = (8)'0'B;
FASE: PUT EDIT('ENTER RECORD CODE') (COL(1),A);
PUT SKIP(2);
BEGIN;
ON CONVERSION BEGIN;
PUT EDIT('RECORD CODE WAS IN ERROR',
'IT MUST BE 7 DIGITS.')
(COL(2),A);
PUT SKIP(2);
GO TO FASE;
END;
GET EDIT(ANS) (COL(1),A(80));
IF SUBSTR(ANS,1,1) = 'S' & CTYPE = 'S'
THEN CALL SIGCHG;

```



```

ELSE;
IF SUBSTR(ANS,1,1) = 'N' | SUBSTR(ANS,1,1) = 'Y'
THEN GO TO LAB100;
GET STRING(ANS) EDIT(ASECODE) (P'9999999');
IF ASECODE < '0010000' THEN DO;
PUT EDIT('YOUR RECORD CODE MUST BE GREATER THAN',
' 0010000 ') (COL(1),A,A);
PUT SKIP(2);
GO TO LAB2(1);
END;
END;
CARD.ASE_CODE = ASECODE;
IF IDUP = 1 THEN GO TO NDUP;
IF SUBSTR(ASECODE,4,4) = '0000' THEN DO;
II = 1;
CTYPE = 'H';
END;
ELSE DO;
LAB11: PUT EDIT('ENTER ADDITION TYPE') (COL(1),A);
IF PROMPT = 'Y' | PROMPT1 = 'Y' THEN
PUT EDIT(
'D - SYSTEM DEVELOPMENT STEP',
'R - SYSTEM REMARK',
'N - NONE')
(COL(5),A);
PUT SKIP(2);
PROMPT1 = 'N';
GET EDIT(CTYPE,FILL) (COL(1),A(1),A(5));
ICOUNT = 0;
IF CTYPE = 'N' THEN GO TO LAB100;
II = INDEX('HDRN',CTYPE);
IF II = 0 THEN DO;
PROMPT1 = 'Y';
GO TO LAB11;
END;
END;
NLABEL = II;
CALL ER_PAGE;
CALL ADD(NLABEL,LINE);
IDUP = 1;
NDUP:WRITE FILE(DIREC) FROM(LINE) KEYFROM(ASECODE);
IDUP = 0;
CTYPE = SUBSTR(LINE,113,1);
IF CTYPE = 'S' & SUBSTR(ASECODE,4,4) = '0000' THEN
PUT FILE(ACHG) EDIT(HCARD) (B(8),A);
IF PROMPT = 'Y' THEN
PUT EDIT(' ADDITION COMPLETE. READY FOR NEXT ADDITION.')

```

```

(COL(1),A);
PUT SKIP(2);
TC = 'A';
GO TO LAB2(1);
LAB2(2): /* DELETE A RECORD */
TC = 'D';
PUT EDIT('ENTER RECORD CODE') (COL(1),A);
PUT SKIP(2);
GET EDIT(ANS) (COL(1),A(80));
IF SUBSTR(ANS,1,1) = 'S' & CTYPE = 'S' THEN CALL SIGCHG;
IF SUBSTR(ANS,1,1) = 'N' | SUBSTR(ANS,1,1) = 'Y'
THEN GO TO LAB100;
ASECODE = SUBSTR(ANS,1,7);
IF ASECODE = '0010000' THEN DO;
PUT EDIT('YOU CANNOT DELETE RECORD 0010000',
'YOU CAN CHANGE RECORD 0010000 WITH ',
'THE CHANGE COMMAND.',
'I WILL ASSUME YOU WANT TO DELETE ',
'SOME OTHER RECORD.')
```

```

(COL(1),A,(2)(COL(1),A,A));
GO TO LAB2(2);
END;
LAB0X: READ FILE(DIREC) INTO(LINE) KEY(ASECODE);
DELETE FILE(DIREC) KEY(ASECODE);
IF PROMPT = 'Y' THEN
PUT EDIT(' DELETION COMPLETE. READY FOR NEXT DELETION.')
```

```

(COL(1),A);
IF SUBSTR(ASECODE,4,4) = '0000' THEN GO TO LAB2(2);
CTYPE = SUBSTR(LINE,113,1);
GO TO LAB2(2);
LAB2(3): /* CHANGE A RECORD */
TC = 'C';
PUT EDIT('ENTER RECORD CODE') (COL(1),A);
PUT SKIP(2);
GET EDIT(ASECODE) (COL(1),A(7));
IF ASECODE = 'N' | ASECODE = 'S' THEN GO TO LAB100;
READ FILE(DIREC) INTO(LINE) KEY(ASECODE);
II = 0;
IF SUBSTR(ASECODE,4,4) = '0000' THEN II = 1;
ELSE IF SUBSTR(LINE,113,1) = 'S' THEN II = 2;
IF SUBSTR(LINE,113,1) = 'R' THEN II = 3;
GO TO LAB3(II);
LAB3(0): PROMPT1 = 'Y';
GO TO LAB100;
LAB3(1): /* CHANGE HEADING INFORMATION */
PUT EDIT('ENTER HEADING ITEM TO BE CHANGED')
(COL(1),A);
```

```

IF PROMPT = 'Y' | PROMPT1 = 'Y' THEN
PUT EDIT('M - AIRCRAFT SYSTEM',
'T - SYSTEM TITLE',
'D - AS OF DATE',
'E - START YEAR FOR REPORT',
'R - NUMBER OF YEARS IN REPORT',
'A - APPLICATION CODE',
'N - NONE')
(COL(5),A);
PUT SKIP(2);
GET EDIT(CTYPE,FILL) (COL(1),A(1),A(5));
PROMPT1 = 'N';
IF CTYPE = 'N' THEN GO TO LAB3(4);
NLABEL = INDEX('MTDERAN',CTYPE);
IF NLABEL = 0 THEN DO;
PROMPT1 = 'Y';
GO TO LAB3(1);
END;
CALL CHANGE(NLABEL,LINE,FIN);
IF FIN = 'N' | FIN = 'S' THEN GO TO LAB3(4);
GO TO LAB3(1);
LAB3(2): /* CHANGE STEP INFORMATION */
CH_TY:PUT EDIT('ENTER INFORMATION TYPE TO BE CHANGED') (COL(1),A)
IF PROMPT = 'Y' | PROMPT1 = 'Y' THEN DO;
PROMPT1 = 'N';
PUT EDIT(
(COL(5),A);
END;
PUT SKIP(2);
GET EDIT(ANS) (COL(1),A(80));
IF SUBSTR(ANS,1,1) = 'N' THEN GO TO LAB3(4);
IF INDEX('TDAE',SUBSTR(ANS,1,1)) = 0 THEN DO;
PROMPT1 = 'Y';
GO TO CH_TY;
END;
IF SUBSTR(ANS,1,1) = 'A' THEN DO;
NLABEL = 6;
CHG_EL:CALL CHANGE(NLABEL,LINE,FIN);
IF FIN = 'N' | FIN = 'S' THEN GO TO LAB3(4);
GO TO LAB3(2);
END;
IF SUBSTR(ANS,1,1) = 'E' THEN DO;
NLABEL = 12;
GO TO CHG_EL;
END;
IF SUBSTR(ANS,1,1) = 'T' THEN IY = 1;
ELSE IY = 2;

```

```

IF TYPE_NUMBER < 0 THEN IX = 1;
ELSE IX = 3;
NLABEL = 5 + IX + IY;
CALL CHANGE(NLABEL,LINE,FIN);
IF FIN = 'N' | FIN = 'S' THEN GO TO LAB3(4);
GO TO LAB3(2);
LAB3(3):/* CHANGE A REMARK */
PUT EDIT('ENTER ITEM TO BE CHANGED')
(COL(1),A);
IF PROMPT = 'Y' | PROMPT1 = 'Y' THEN
PUT EDIT('R - REMARK',
'A - APPLICATION CODE',
'N - NONE')
(COL(5),A);
PUT SKIP(2);
GET EDIT(ANS) (COL(1),A(80));
IF SUBSTR(ANS,1,1) = 'N' THEN GO TO LAB3(4);
ELSE;
IF SUBSTR(ANS,1,1) = 'A' THEN NLABEL = 6;
ELSE NLABEL = 11;
CALL CHANGE(NLABEL,LINE,FIN);
IF FIN = 'N' | FIN = 'S' THEN GO TO LAB3(4);
GO TO LAB3(3);
LAB3(4): /* CHANGE COMPLETE */
TEMP_FLAG = CARD.DELETE_FLAG;
SIGCHGN: PROC;
DCL CHG FILE ENV(F(380,38));
PUT EDIT('WAS THIS A SIGNIFICANT CHANGE ?')
(COL(1),A);
PUT SKIP(2);
GET EDIT(ANS) (COL(1),A(80));
IF SUBSTR(ANS,1,1) = 'Y' THEN DO;
ACODE = SUBSTR(ASECODE,1,3);
CTIT1 = DEVEL_STEP.TITLE;
IF TC = 'S' THEN CTIT1 = (28) ' ';
DT = CUR_DATE;
PUT FILE(CHG) EDIT(CHGN_RECORD) (COL(1),(4)A);
END;
END SIGCHGN;
REWRITE FILE(DIREC) FROM(LINE) KEY(ASECODE);
PUT FILE(ACHG) EDIT(HCARD) (8(8),A);
IF PROMPT = 'Y' THEN
PUT EDIT('CHANGE COMPLETE. READY FOR NEXT CHANGE.')
(COL(1),A);
PUT SKIP(2);
IF FIN = 'S' THEN CALL SIGCHGN;
ELSE;

```



```

GO TO LAB2(3);
LAB2(4):; /* SLIP A SCHEDULE */
PUT EDIT('ENTER SYSTEM CODE ') (COL(1),A);
PUT SKIP(2);
TC = 'S';
GET EDIT(ASECODE) (COL(1),A(7));
IF SUBSTR(ASECODE,1,1) = 'N' THEN GO TO LAB100;
ASECODE = SUBSTR(ASECODE,1,3) || '0000';
READ FILE(DIREC) INTO(LINE) KEY(ASECODE);
ASECODE1 = ASECODE;
CALL SSLIP(ASECODE);
ASECODE = ASECODE1;
GO TO LAB100;
LAB2(5):; /* COPY A RECORD FROM ONE FILE TO ANOTHER */
PUT EDIT(' COPY DOES NOT WORK IN THIS VERSION OF SCHEDULE')
(COL(5),A);
PUT SKIP(2);
PROMPT1 = 'Y';
GO TO LAB100;
LAB2(6):; /* LIST A RECORD OR A NUMBER OF RECORDS */
PUT EDIT('ENTER LIST TYPE') (COL(1),A);
IF PROMPT = 'Y' | PROMPT1 = 'Y' THEN
PUT EDIT('A - SYSTEM HEADING RECORDS',
'S - ALL RECORDS OF SYSTEM X',
'P - SUBSET OF ALL RECORDS OF SYSTEM X',
'N - NONE') (COL(5),A);
PUT SKIP(2);
GET EDIT(LTYPE,FILL) (COL(1),A(1),A(5));
PROMPT1 = 'N';
IL = INDEX('ASPN',LTYPE);
CALL ER_PAGE;
GO TO LAB6(IL);
LAB6(0):; /* CODE NOT FOUND */
PUT EDIT('INVALID LIST CODE') (COL(1),A);
PUT SKIP(2);
PROMPT1 = 'Y';
GO TO LAB2(6);
LAB6(1):; /* LIST ALL SYSTEM HEADING RECORDS */
L1 = 0; L2 = 0;
DO WHILE('1'B);
L1 = L1 + 1;
ASECODE = CHAR(L1) || CHAR(L2);
PUT SKIP(2);
READ FILE(DIREC) INTO(LINE) KEY(ASECODE);
CALL LLIST(LINE);
END;
GO TO LAB2(6);

```

```

LAB6(2):/* LIST ALL RECORDS FOR SYSTEM X */
PUT EDIT('ENTER SYSTEM CODE') (COL(1),A);
PUT SKIP(2);
GET EDIT(ASECODE) (COL(1),A(7));
ASECODE = SUBSTR(ASECODE,1,3)||'0000';
LAB81: READ FILE(DIREC) INTO(LINE) KEY(ASECODE);
CALL LLIST(LINE);
DO WHILE('1'B);
READ FILE(DIREC) INTO(LINE) KEYTO(ASECODE);
IF SUBSTR(ASECODE,4,4) = '0000' THEN GO TO LAB100;
CALL LLIST(LINE);
END;
GO TO LAB2(0);
LAB6(3):/* LIST SUBSET OF ALL RECORDS FOR SYSTEM X */
PUT EDIT('ENTER SYSTEM CODE') (COL(1),A);
PUT SKIP(2);
GET EDIT(ASECODE) (COL(1),A(7));
ASECODE = SUBSTR(ASECODE,1,3)||'0000';
LAB631: PUT EDIT('ENTER APPLICATION CODE') (COL(1),A);
IF PROMPT = 'Y' | PROMPT1 = 'Y' THEN DO;
IF JENNY1 = 'P' THEN
PUT EDIT('B - BUDGET','E - PROJECT ENGINEER',
'H - PROJECT MANAGER','O - 104 REPORT',
'P - PROCUREMENT','Q - PRODUCT ASSURANCE',
'R - RECAP','S - PROGRAM SUMMARY',
'T - TEST ')
(COL(5),A);
IF JENNY1 = 'R' THEN
PUT EDIT('C -COMPTROLLER','B - PLANS & ANALYSIS',
'P - PROCUREMENT','E - DEVELOPMENT & ENGINEERING',
'F - ASTIO','T - TRADOC','S - TECOM',
'Q - PRODUCT ASSURANCE')
(COL(5),A);
END;
PUT SKIP(2);
GET EDIT(APLICODE2) (COL(1),A(12));
CALL ER_PAGE;
IF APLICODE2 = ' ' THEN GO TO LAB631;
LSTR = INDEX(APLICODE2,' ') -1;
APLICODE = SUBSTR(APLICODE2,1,LSTR);
READ FILE(DIREC) INTO(LINE) KEY(ASECODE);
CALL LLIST(LINE);
DO WHILE('1'B);
READ FILE(DIREC) INTO(LINE) KEYTO(ASECODE);
IF SUBSTR(ASECODE,4,4) = '0000' THEN GO TO LAB2(6);
IF (INDEX(SUBSTR(LINE,9,11),APLICODE) ~= 0 & JENNY3 = 'N') |
(INDEX(SUBSTR(LINE,9,LSTR),APLICODE) ~= 0 & JENNY3 = 'P')

```

```
THEN CALL LLIST(LINE);  
END;  
GO TO LAB2(0);  
LAB6(4):/* NO LISTS REQUIRED */  
GO TO LAB100;  
LAB2(7): /* FIND A RECORD */  
PUT EDIT(' FIND DOES NOT WORK IN THIS VERSION OF SCHEDULE.')(COL(5),A);  
PUT SKIP(2);  
GO TO LAB100;  
LAB2(8): /* NO MORE CHANGES REQUIRED */  
CLOSE FILE(DIREC);  
RETURN;  
ER_PAGE:PROC;  
CALL NEWPAG;  
CALL TSEND;  
RETURN;  
END ER_PAGE;  
END UPDATE;
```

```

WINDOW:PROC;
DCL ANS CHAR(80) VAR;
DCL (LLIMIT,ULIMIT,STYR,NOYR) FIXED DEC(5,2) EXTERNAL;
DCL DUMMY FIXED DEC(5,2);
DCL (LOWER,UPPER) CHAR(9);
DCL LAB(0:4) LABEL;
I = 0;
ON CONVERSION BEGIN;
PUT EDIT('YOUR INPUT WAS IN ERROR')
(COL(1),A);
PUT EDIT(ONSOURCE)(COL(1),A);
PUT EDIT('PLEASE RE-ENTER')
(COL(1),A);
PUT SKIP(2);
REVERT CONVERSION;
GO TO LAB(I);
END;
LAB(0):PUT EDIT('ENTER LOWER TIME LIMIT')
(COL(1),A);
CALL FORMD;
FORMD:PROC;
PUT EDIT(' / * _ * _ / ') (COL(1),A);
PUT SKIP(2);
CALL REVERSE;
GET EDIT(ANS) (COL(1),A(80));
RETURN;
END FORMD;
GET STRING(ANS) EDIT(LOWER) (X(1),P'99AAAAA99');
CALL FDATE(LOWER,DUMMY,0);
STYR = DEC(SUBSTR(LOWER,8,2));
LLIMIT = DUMMY - 1;
I = 1;
LAB(1):PUT EDIT('ENTER UPPER TIME LIMIT')
(COL(1),A); PUT SKIP(2);
CALL FORMD;
GET STRING(ANS) EDIT(UPPER) (X(1),P'99AAAAA99');
CALL FDATE(UPPER,DUMMY,0);
NOYR = DEC(SUBSTR(UPPER,8,2)) - STYR + 1;
IF NOYR < 1 THEN NOYR = 1;
ULIMIT = DUMMY - 1;
LAB(2):;
RETURN;
END WINDOW;

```


Appendix B
Overlay Listing

JOB 0042

//LSECD=JOB L2IOAL00205.50...MIKE.GARSIK.CLASSM

***OUTE PRINT REMOTE2 ZH 2 6 0 0 00000020

***OUTE PRINT REMOTE2 ZH 2 6 0 0 00000030

***OUTE PRINT REMOTE2 ZH 2 6 0 0 00000040

//STEP1 EXEC PGM=IEFGEN,REGION=500 00000050

//SYSPRINT DD SYSOUT=A 00000060

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000070

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000080

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000090

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000100

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000110

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000120

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000130

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000140

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000150

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000160

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000170

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000180

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000190

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000200

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000210

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000220

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000230

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000240

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000250

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000260

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000270

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000280

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000290

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000300

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000310

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000320

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000330

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000340

//SYSDA DD DSN=LPMASMG.X.MASTER.LINKLIST,UNIT=DISK,DISP=SHR 00000350

DATA SET UTILITY - GENERATE

PAGE 0001

PROCESSING FINISHED AT EOD

NAME	ORIGIN	LENGTH	SEC. NO.	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION
IMESAP	•	1080	03C	IMESADA	1080	IMESAPC	10A2	IMESAPD	10AA	IMESAPA	10B2
				IMESAPB	10BA	IMESAPF	10C2	IMESADB	10CA	IMESADG	10D2
				IMESAPC	10DA	IMESAFB	10E2	IMESAFB	10EA	IMESAFD	10F2
				IMESAKA	10FA	IMESAFD	10G2	IMESARC	25F4	IMESADD	2700
				IMESAFF	2742						
IMECSU	•	2040	06	IMECSO	2040						
IMEIDA	•	2070	1AA	IMEIDAA	2070	IMEIDAB	2072	IMEIDAC	2074	IMEIDAD	2076
				IMEIQAT	2060						
IMEIDB	•	2AEU	1E0	IMEIDBA	2AE0	IMEIDBB	2AE0	IMEIDBC	2AF0	IMEIDBD	2AF0
				IMEIDBE	2B00	IMEIDBT	2BE0				
IMEIDC	•	2CCU	12A	IMEIDCA	2CC0	IMEIDCB	2CC2	IMEIDCC	2CC6	IMEIDCT	2D7A
IMEIDT	•	2JEG	104								
TJEND	•	2F80	110								
IMEUCN	•	3000	256								
IMEUMA	•	32EU	08	IMEUCNA	3098	IMEUCNB	309A				
				IMEUMAA	32F0						
IMEJXS	•	30E0	00	IMEJXSI	33E0	IMEJXSJ	3404				
IMESRC	•	3450	1A2	IMESRCA	3450	IMESRCB	3452	IMESRCC	3454	IMESRCD	3456
				IMESRCE	3450	IMESRCF	345A				
IMESSE	•	35F0	84	IMESSE0	35E4						
IMEVPU	•	3080	105	IMEVPOA	3080						
IMEVPE	•	3700	270	IMEVPEA	3700						
IMEWSC	•	3A30	AC	IMEWSCA	3A30						
TOUTPT	•	3AEU	13A								
IMEKCA	•	3C20	110	IMEKCA	3C20	IMEKCB	3C22				
IMEUPB	•	3030	04	IMEUPBA	3030	IMEUPBB	3090				
IMEVFA	•	3F10	160	IMEVFAA	3E10						
IMEVEL	•	3E00	20	IMEVELA	3F00						
IMEVFE	•	3F00	20	IMEVFEA	3F00						
IMEVKE	•	3F00	070	IMEVKEA	3F00						
IMEVPA	•	4050	1E0	IMEVPAA	4050						
IMEVPE	•	4030	50	IMEVPFA	4030						

NAME	ORIGIN	LENGTH	SEG. NO.	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION
IMEVPA	• 4880	229	1	IMEVPA	4880				
IMEVSE	• 4880	150	1	IMEVSE	4800	IMEVSEB	4802		
IMECSM	• 4810	114	1	IMECSMH	4810	IMECSML	4810	IMECSMF	4830
				IMECSMR	4810			IMECSMV	4840
IMECSV	• 4028	84	1	IMECSVA	4028				
IMEODA	• 4000	238	1	IMEODAA	4000	IMEODAB	4002		
IMELOE	• 5020	04	1	IMELOEA	5020				
IMELOA	• 5100	108	1	IMELONA	5100				
IMELOP	• 5200	434	1	IMELOPA	5200	IMELOPB	5204	IMELOPC	5204
				IMELOPE	5504			IMELOPD	5504
IMEUPA	• 5640	08	1	IMEUPAA	5640	IMEUPAB	5640		
IMEVFA	• 5728	66	1	IMEVFDA	5728				
IMEVSK	• 5790	120	1						
IMEVSK	• 5880	104	1	IMEVSKR	5880	IMEVSKL	5880	IMEVSKM	5880
IMEVTE	• 5888	104	1	IMEVTEA	5888				
IMEVNC	• 5050	282	1	IMEVNCA	5050				
IMEVNU	• 5008	128	1	IMEVNUA	5008	IMEVNUB	5008		
IMEVPA	• 6030	142	1	IMEVPBA	6030				
IMEVAT	• 6108	180	1						
IMEVAG	• 6368	186	1						
IMEVFB	• 6520	246	1						
IMECSI	• 6768	142	1	IMECSIA	6768				
IMEKCA	• 6810	614	1	IMEKCSA	6810				
IMEVSO	• 6828	08	1	IMEVSUA	6828				
IMEVFB	• 6000	00	1	IMEVFB	6000				
IMEVKB	• 6000	304	1	IMEVKB	6000				
IMEVSB	• 7000	00	1	IMEVSB	7000				
IMELOI	• 7300	078	1	IMELOIA	7300	IMELOIB	7302	IMELOIC	7304
MUVABS	• 7048	150	1	IMELOIA	7300	IMELOIB	7302	IMELOIC	7304

NAME ORIGIN LENGTH SEQ. NO. NAME LOCATION NAME LOCATION NAME LOCATION

IMNSURIO	C224	168	1	SURT	C920			
IMNEAP	C888	180	1	EXP	C488			
IMNECUMH	C436	180	1	IBCOM	C464	F010CSW	C020	INTSMITH DB9E
IMNLOH2	D0C4	715	1	SEQUASD	DF66			
IMNEARM	E2EU	5FC	1	ERRON	E2E0	IMNERME	E2F8	
PLTCHK	E0EU	2D4	1					
RASA	E888	100	1					
IMNECVIH	FCB4	84E	1					
IMNEFNTH	F828	546	1	AUCONH	EC88	FCVAOUTP	E062	FCVLOUTP E0F2 FCV2OUTP EF4E
IMNEFIOS	FD70	FF8	1	FCVLOUTP	E2E6	FCVLOUTP	E3E8	FCVLOUTP E532
IMNFIUS2	10D8	588	1	ARTIM	E828	AUJSMITH	E8C4	
IMNUOPT	1120	318	1	F10CSH	E070	F10CSH	E076	
IMNFCUNI	11638	2E5	1					
IMNFCUNU	11920	4A2	1	FCUNIM	11638			
IMNETWCH	110C8	2A6	1	FCUNIM	11920			
IMNUATBL	12070	638	1	IMNIRCH	110C8	ERRIRA	110D0	
IMNETEN	126A8	198	1					
NUYK	12840	3	1	FTEN#	126A8			
STYK	12848	3	1					
ULIMIT	12850	3	1					
LLIMIT	12858	3	1					
NKARY	12860	3	1					
NSARY	12868	3	1					
TALINA	12870	FD	1					
BPPCOM	12960	140	1					
SENTAB	12A80	60	1					
GRAP	12B00	2F0	2					
GRAPA	124EU	194	2					
RUATE	12F88	584	2					
BUAIEA	13540	8H	2					
REVERSE	13600	84	2					
REVERSE	13688	4F	2					
FUATE	13808	E34	2					
BUAIEA	14510	392	2					
SENTAB	146A8	48	2					

[illegible]

MINUCo	14850	54	3
INDUMA	14FD0	228	3

NEW	165EU	34DC	4
0000NEWA	170DU	C57	4
SID	18A24	38	6
NEW1	18A6U	16FC	4
0000NEW1A	1A160	773	4

UPDATE	140F0	34E0	5
BURDAIEA	170DU	104E	5
CMG	10E2U	38	5
ACHG	10E58	30	5
ENTAB	19E90	3C	5

ADU	18500	1884	6
ADLDA	1AA56	855	6

CHANGE	18500	3484	7
CHANGEA	10386	ADD	7

SLIP	18E00	F6C
00SLIPA	19E40	044

LIST	DATE	SEC
00LLISTA	19CCO	301
		9

FURN	16050	6284	10
000FOMA	18070	532	10

19730	104	10
19564	166	10
19730	104	10

CAX15	•	1A568	146	10
OKWAB5	•	1A698	104	10

LINEF	•	1AB54	1CA	10
LINE=DI	•	1AU26	14C	10

SLURSA	•	18160	28F	10
ScTMRG	•	18420	142	10

NAME	ORIGIN	LENGTH	NAME	ORIGIN	LENGTH	NAME	ORIGIN	LENGTH
00PLATEM	9U	4	CHG	94	4	ACHG	98	4
00RADDLC	AD	4	ACHANGED	AA	4	ACHANGED	AC	4
00SSLIPD	BU	4	00SSLIPC	B4	4	00LLISTB	B6	4
00RFORMC	CO	4	00RFORMC	C4	4	00RFORMB	BC	4
01TYREPTD	DO	4	01TYREPTD	D4	4	01TYREPTC	CC	4
02RURISU	EO	4	02RURISE	E4	4	02RURISC	DC	4
REPORTSM	FO	4	REPORTSI	F4	4	REPORTSC	EC	4
REPORTSL	LOU	4	REPORTSL	LO4	4	REPORTSK	FC	4
IMEULW2	110	4	IMEULW3	114	4	IMEULW1	10C	4
IMEULLA	120	4	IMEULWA	124	4	IMEULNE	11C	4
IMEUCFL	130	8	IMEGADC	138	4	IMEGADP	12C	4
IMEULW8	14C	4	IMEUSLA	150	4	IMEGXLV	144	8
IMEURTC	15C	4	IMEUSFC	160	4	IMEGSLM	158	4

TOTAL LENGTH OF PSEUDO REGISTERS 164

ENTRY ADDRESS 060

TOTAL LENGTH 20158

00000000 DOES NOT EXIST BUT HAS BEEN ADDED TO DATA SET

DIAGNOSTIC MESSAGE DIRECTORY

180461 WARNING - SYMBOL PRINTED IS AN UNRESOLVED EXTERNAL REFERENCE; NCAL WAS SPECIFIED, OR THE REFERENCE WAS MARKED FOR RESTRICTED NO-CALL OR NEVERCALL.

Appendix C

Maintenance Subroutine Listings


```

'LAUGEVA,COMPRS,CNTL'
00010 //LAUGE1B JOB (3E02,L060,5,4),'INDEX SEQ JOB',CLASS=K,
00020 // NOTIFY=LAUGEVA,REGION=100K
00021 //ROUTE PRINT REMOTES
00030 //STEP1 EXEC PGM=IEBISAM,PARM=COPY
00040 //SYSPRINT DO SYSOUT=A
00050 //SYSUT1 DO DSN=L.WILL INDEX,DISP=(OLD,KEEP),UNIT=DISK,
00060 // DCB=(OPTCD=LIY,DSORG=IS,RECFM=FB,LRECL=114,BLKSIZE=1140)
00070 //SYSUT2 DO DSN=TEMP2.DATA.SET<INDEX>,DISP=(NEW,CATLG,DELETE),
00080 // DCB=(OPTCD=LIY,DSORG=IS,RECFM=FB,BLKSIZE=1140,LRECL=114),
00090 // SPACE=(CYL,SPACER),VOL=SER=XXXX,UNIT=DISK
00100 // DO DSN=TEMP2.DATA.SET<PRIME>,DISP=(NEW,KEEP,DELETE),
00110 // DCB=(OPTCD=LIY,DSORG=IS,RECFM=FB,LRECL=114,BLKSIZE=1140),
00120 // SPACE=(CYL,SPACER),VOL=SER=XXXX,UNIT=DISK
00130 // DO DSN=TEMP2.DATA.SET<OVLFLOW>,DISP=(NEW,KEEP,DELETE),
00140 // DCB=(OPTCD=LIY,DSORG=IS,RECFM=FB,LRECL=114,BLKSIZE=1140),
00150 // SPACE=(CYL,SPACER),VOL=SER=XXXX,UNIT=DISK
00160 //STEP2 EXEC PGM=IEBISAM,PARM=COPY
00170 //SYSPRINT DO SYSOUT=A
00180 //SYSUT2 DO DSN=L.WILL INDEX,DISP=(OLD,KEEP),UNIT=DISK,
00190 // DCB=(OPTCD=LIY,DSORG=IS,RECFM=FB,LRECL=114,BLKSIZE=1140)
00200 //SYSUT1 DO DSN=TEMP2.DATA.SET<INDEX>,DISP=(OLD,DELETE,DELETE),
00210 // DCB=(OPTCD=LIY,DSORG=IS,RECFM=FB,BLKSIZE=1140,LRECL=114),
00220 // SPACE=(CYL,SPACER),UNIT=DISK
00230 // DO DSN=TEMP2.DATA.SET<OVLFLOW>,DISP=(OLD,DELETE,DELETE),
00240 // DCB=(OPTCD=LIY,DSORG=IS,RECFM=FB,BLKSIZE=1140,LRECL=114),
00250 // SPACE=(CYL,SPACER),VOL=SER=XXXX,UNIT=DISK
00260 // DO DSN=TEMP2.DATA.SET<PRIME>,DISP=(OLD,DELETE,DELETE),
00270 // DCB=(OPTCD=LIY,DSORG=IS,RECFM=FB,LRECL=114,BLKSIZE=1140),
00280 // SPACE=(CYL,SPACER),VOL=SER=XXXX,UNIT=DISK
00290 //
READY

```

```

'LAUGEUA.SCH.CLIST<COMPRS>'  

00010 PROC 1 NAME DISK<DFSE01> REMOTE<REMOTE2> SPACE<1>  

00020 CD MEL.CNTL  

00030 E 'LAUGEUA.COMPRS.CNTL' CNTL  

00040 C 10 290 /L.WILL/&NAME./ALL  

00050 C 10 290 /XXX/&DISK./ALL  

00051 C 10 290 /SPACER/&SPACE./ ALL  

00053 C 21 /REMOTE8/&REMOTE./  

00060 S MEL.CNTL  

00070 END  

00080 SUBMIT MEL.CNTL  

00090 END  

READY

```

```

'LAUGEVA.SCH.CLIST<DUMP>'
00010 PROC 1 NAME DISP<OLD> CATLG<KEEP> REMOTE<REMOTE8> LABEL<>
00020 CF DAX 'LAUGEVA.COMPRS.CNTL'>
00030 CD WEL2.CNTL
00040 E 'LAUGEVA.COMPRS.CNTL' CNTL
00041 00022 /*SETUP TS0168
00050 D 70 290
00070 C 50 /*L.WILL/&NAME /
00090 C 21 /*REMOTE8/&REMOTE /
00100 C 30 /*COPY/UNLOAD/
00110 00070 /*SYSUT2 DD DSN=&NAME..BACK..INDEX,UNIT=TAPE9,
00111 000080 // DCB=(LRECL=80,BLKSIZE=640,RECFM=FB),
00112 00090 // DISP=(&DISP.,&CATLG.),LABEL=(&LABEL.,SL)
00113 00100 /*
00114 00110 //
00140 S WEL2.CNTL
00150 END
00170 CF DAX 'LAUGEVA.COMPRS.CNTL'>
00180 END
READY

```

```

'LAUGEVA SCH.CLIST(LIST)'
00010 PROC 1 NAME INDEX(>) REMOTE(REMOTE2>)
00011 CF DAX 'LAUGEVA.COMPRS.CNTL'>
00020 CO WELL1.CNTL
00030 E 'LAUGEVA.COMPRS.CNTL' CNTL OLD
00040 D 70 290
00051 C 50 /L.WILL INDEX/&NAME...INDEX&INDEX./
00060 C 21 /REMOTE8/&REMOTE./
00070 C 30 /COPY/'PRINTL,N'./
00080 00070 //SYSUT2 DD SYSOUT=A
00090 00080 /*
00100 00090 //
00101 S WELL1.CNTL
00110 END
00111 SUBMIT WELL1.CNTL
00112 CF DAX 'LAUGEVA.COMPRS.CNTL'>
00120 END
READY

```



```

'LAUGEVA.SCH.CLIST<RESTORE>'
00010 PROC 1 NAME SPACE(1) DISK<DFSE01>
00020 CF DAX 'LAUGEVA.CMPRS.CNTL'>
00030 CD WEL3.CNTL
00040 E 'LAUGEVA.CMPRS.CNTL' CNTL
00050 00022 /XSETUP TS0160
00060 D 160 200
00070 C 30 /COPY/LOAD/
00080 C 50 /L.WILL.INDEX/&NAME...BACK.INDEX/
00090 C 50 /DISK,/TAPE9/
00100 D 60
00110 C 70 150 /SPACER/&SPACE./ ALL
00120 C 70 150 /TEMP2.DATA.SET/&NAME...INDEX/ ALL
00130 C 70 150 /XXXX/&DISK./ ALL
00140 S WEL3.CNTL
00150 END
00160 CF DAX 'LAUGEVA.CMPRS.CNTL'>
00170 END
READY

```

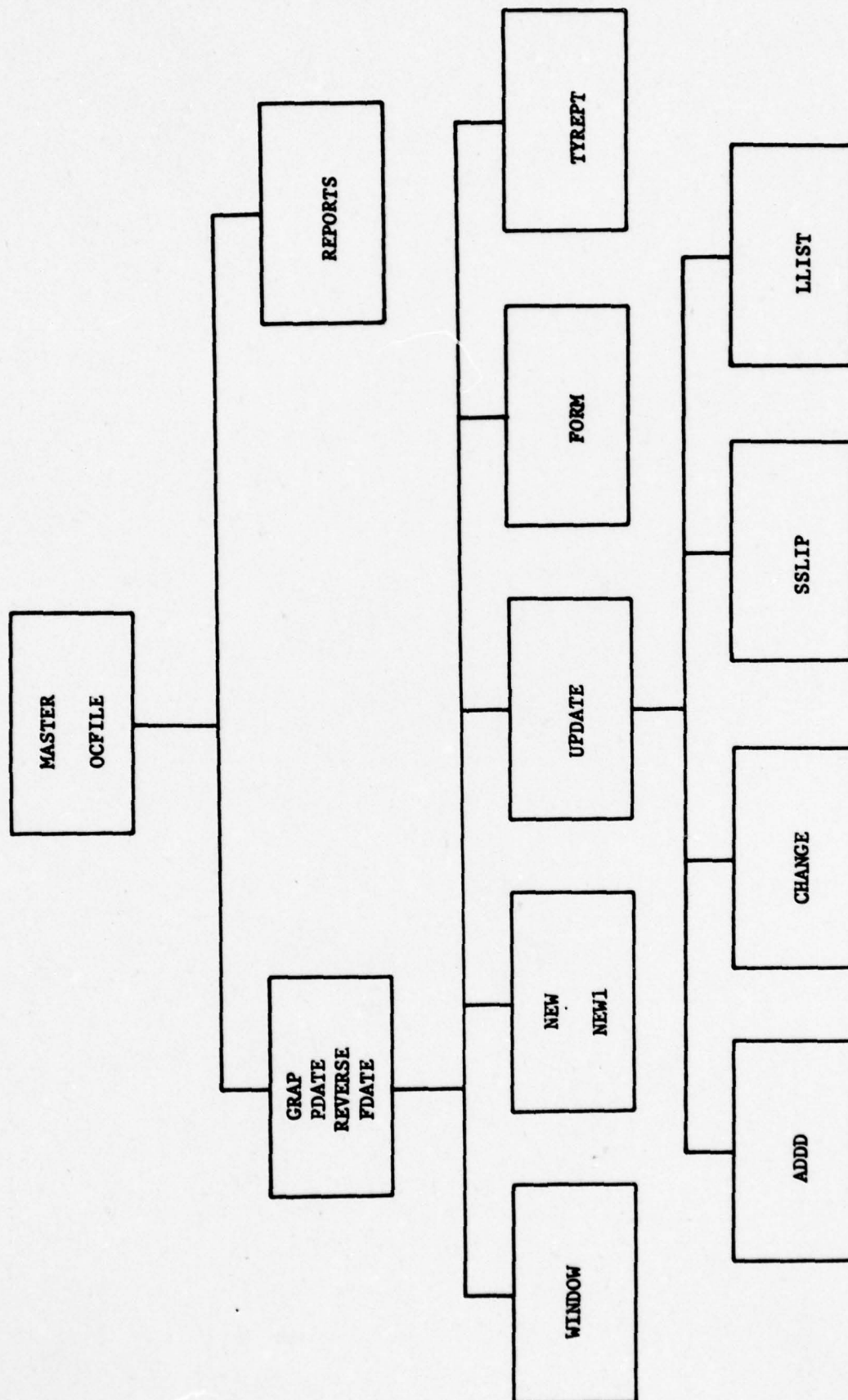


Figure 1.1 Single Region Overlay Tree Structure

ADDD.PLI
FORM.FORT
GRAP.PLI
LLIST.PLI
MASTER.PLI
NEW.PLI
NEW1.PLI
REPORTS.PLI
S.CHANGE.PLI
SSLIP.PLI
TYREPT.PLI
UPDATE.PLI
WINDOW.PLI
A.OCFILE.PLI
FDATE.PLI
RDATE.PLI
REVERSE.PLI

Figure 3.1 SCHEDULE Subroutines

```

SCHEDULE.CLIST
00010 PROC 0 LNK 33) LSZ(133) VERS(X) JEN1(P) JEN2(N) JEN3(N) JEN4(N) JEN5
(N)
00020 TERM LINES(&LN.) LINESIZE(&LSZ.)
00030 CF DAX 'LPMSG.INDEX'
00040 CF DAX 'LPMSG.DATA.DATA'
00050 CF F(FT04F001,FT05F001,FT06F001)
00060 CF F(STD)
00070 CF F(SYSIN)
00080 CF F(DIRECT)
00090 CF DAX 'LPMSG.&VERS..MASTER.LOAD(RUN)')
00100 ALLOC F(STD) DAX 'LPMSG.STDIT.DATA') SHR
00110 ALLOC F(ACHG) DAX 'LPMSG.ACHGN') SHR
00120 ALLOC F(SYSPRINT) DAX(*)
00130 ALLOC F(SCRATCH) DAX 'LPMSG.DATA.DATA') SHR
00140 ALLOC F(FT04F001) DAX 'LPMSG.DATA.DATA') SHR
00150 ALLOC F(DIRECT) DAX 'LPMSG.INDEX') MOD
00160 ALLOC DAX(*) F(SYSIN)
00170 ALLOC DAX(*) F(FT05F001)
00180 ALLOC DAX(*) F(FT06F001)
00190 CALL 'LPMSG.&VERS..MASTER.LOAD(RUN)' '&JEN1.&JEN2.&JEN3.&JEN4.&JEN
5.'
00200 END

```

Figure 3.2 SCHEDULE.CLIST Listing

<u>ARGUMENT NAME</u>	<u>ARGUMENT VALUE</u>	<u>RESULT</u>
Jenny 1	P	Prints Project Manager Application Code Compting List
Jenny 1	R	Prints R&D Command Application Code Prompting List
Jenny 1	N	Disable Application Code Prompting List Printing; Default Value
Jenny 2	C	Enable Cross System Reporting
Jenny 2	N	Disable Cross System Reporting Default Value
Jenny 3	P	Enable Position Dependent Application Code (not available)
Jenny 3	N	Disable Position Dependent Application Code Default Value
Jenny 4	M	Enable Lines Per Page
Jenny 4	N	Disable Lines Per Page; Default Value
Jenny 5	W	Enable Windowing
Jenny 5	N	Disable Windowing; Default Value

Figure 3.3 SCHEDULE Passed Argument Values

```

STDTIT DATA
00010 ** CONCEPT *****
00020 ROC
00030 BOI
00040 PQQPRI
00050 DP
00060 BCE
00070 FREQ ALDC (PRLM)
00080 REPTABLISH TIMG
00090 CRITICAL ISSUE/TEST MATRIX
00100 DT/OT TEST CRITERIA
00110 DT/OT INOPT EVAL PLAN
00120 INIT TOP (OT I)
00130 INIT OTP (OT I)
00140 INIT TOP (OT I)
00150 INIT OTP (OT I)
00160 CTP-I (DRAFT)
00170 CFP (IPR)
00180 CFP (IPR) APPROVAL
00190 ** ADVANCED DEV *****
00200 REVISE CTP-I
00210 DETMN AND FINDING (AD)
00230 SPECS (AD)
00240 STATEMENT OF WORK (AD)
00250 DATA CALL (AD)
00260 DATA REQMTS REVIEW BO (AD)
00270 CONTR DATA REQMTS LIST (AD)
00280 CONTR TEST REQMTS (AD)
00290 PROOTY ENGRG & PLANNING (AD)
00300 PREP EVAL CRITERIA (AD)

```

Figure 3.4 Sample Standard Title Dataset Listing

00310	PREP SECURITY CLSFN GUIDE (AD)
00320	INDPT GOUT COST EST (AD)
00330	SOLE SOURCE DECISION (AD)
00340	RFP (AD)
00350	EVAL CONTR PROPOSALS (AD)
00360	KO TEAM REVIEW (AD)
00370	CONTRACT AWARDS BO (AD)
00380	CONTRACT AWARD (AD)
00390	APPR CONTR TEST PLANS (AD)
00400	FINAL TOP (OT I)
00410	FINAL OTP (OT I)
00420	FINAL TOP (OT I)
00430	FINAL OTP (OT I)
00440	CTP-I (FINAL)
00450	INIT PLAN FOR LOG SUPT
00460	AIRCRAFT BAILED (AD)
00470	SAFETY OF FLT RELEASE (AD)
00480	DT I (EDT-C)
00490	DT I (EDT-C) REPT(PRLM)
00500	DT I (EDT-C) REPT(REVIEW)
00510	DT I (EDT-C) REPT(FINAL)
00520	DT I (ADUT-C)
00530	DT I (ADUT-C) REPT (PRLM)
00540	DT I (ADUT-C) REPT (REVIEW)
00550	DT I (ADUT-C) REPT (FINAL)
00560	DT I (EDT-G)
00570	DT I (EDT-G) REPT (PRLM)
00580	DT I (EDT-G) REPT (REVIEW)
00590	DT I (EDT-G) REPT (FINAL)
00600	DT I (ADUT-G)

Figure 3.4 Sample Standard Title Dataset Listing (cont'd)

00610	DT I	<ADUT-G>	REPT	<PRLM>
00620	DT I	<ADUT-G>	REPT	<REVIEW>
00630	DT I	<ADUT-G>	REPT	<FINAL>
00640	DT I			
00650	DT I	REPT	<PRLM>	
00660	DT I	REPT	<REVIEW>	
00670	DT I	REPT	<FINAL>	
00680	DT I	INDPT	EVAL	
00690	DT I	INDPT	EVAL	
00700	DT I	INDPT	EVAL	REPT
00710	DT I	INDPT	EVAL	REPT
00720	INIT	TDP	<OT II>	
00730	INIT	ODP	<OT II>	
00740	INIT	TDP	<OT II>	
00750	INIT	ODP	<OT II>	
00760	CTP-II	<DRAFT>		
00770	FREQ	ALDC	<DEV>	
00780	VALIDATION	<IPR>		
00790	VALIDATION	<IPR>	APPROVAL	
00800	** ENGR	DEV	*****	
00810	REVISE	CTP II		
00820	DETMN	AND FINDING	<ED>	
00830	PIP			
00850	STATEMENT	OF WORK	<ED>	
00860	DATA	CALL<ED>		
00870	DATA	REQMTS	REVIEW	BO <ED>
00880	CONTR	DATA	REQMTS	LIST <ED>
00890	CONTR	TEST	REQMTS	<ED>
00900	PROOTY	ENCRG	& PLANNING	<ED>
00910	PREP	EVAL	CRITERIA	<ED>

Figure 3.4 Sample Standard Title Dataset Listing (cont'd)

00920	PREP SEC CLSFN GUIDE <ED>
00930	INDPT GOVT COST EST <ED>
00940	SOLE SOURCE DECISION <ED>
00950	RFP <ED>
00960	RFO <ED>
00970	EVAL CONTR PROPOSALS <ED>
00980	KO TEAM REVIEW <ED>
00990	CONTRACT AWARDS BD <ED>
01000	CONTRACT AWARD <ED>
01010	INIT DESIGN REVIEW <ED>
01020	APPR CONTR TEST PLANS <ED>
01030	FINAL TDP <OT II>
01040	FINAL ODP <OT II>
01050	FINAL TDP <OT II>
01060	FINAL ODP <OT II>
01070	CTP-II <FINAL>
01080	PROTO FABRICATION <ED>
01090	REVIEW PROO PLAN <ED>
01100	AIRCRAFT BAILED <ED>
01110	PROTO TRIAL INSTL <ED>
01120	DT II <EDT-C>
01130	DT II <EDT-C> REPT <PRLM>
01140	DT II <EDT-C> REPT <REVIEW>
01150	DT II <EDT-C> REPT <FINAL>
01160	DT II <PQT-C>
01170	DT II <PQT-C> REPT <PRLM>
01180	DT II <PQT-C> REPT <REVIEW>
01190	DT II <PQT-C> REPT <FINAL>
01200	TECH DRAWING PKG REVIEW <ED>
01210	FINAL DESIGN REVIEW <ED>

Figure 3.4 Sample Standard Title Dataset Listing (cont'd)

01220	TECH DATA PKG REVIEW (EO)
01230	INIT PLAN FOR LOG SUPT
01240	NET TEST PERSONNEL
01250	PROV ACCT OF SYS FOR TEST
01260	SAFETY OF FLT RELEASE (EO)
01270	DT II (EDT-G)
01280	DT II (EDT-G) REPT(PRLM)
01290	DT II (EDT-G) REPT(FINAL)
01300	DT II (PQT-G)
01310	DT II (PQT-G) REPT(PRLMN)
01320	DT II (PQT-G) REPT(REVIEW)
01330	DT II (PQT-G) REPT (FINAL)
01340	OT II TEST
01350	OT II REPT (PRLM)
01360	OT II REPT (REVIEW)
01370	OT II REPT (FINAL)
01380	OT II INOPT EVAL
01390	OT II INOPT EVAL
01400	OT II INOPT EVAL (REPT)
01410	OT II INOPT EVAL (REPT)
01420	UPDATE BOI&TOE
01430	UPDATE PLAN FOR LOG SUPT
01440	FINAL REVIEW PROO PROCESS
01450	PROVISIONING CONFERENCE
01460	DRAFT TNG PROGRAM
01470	UPDATE QGPRI
01480	UPDATE TECH DATA PKG
01490	INIT TOP (OT III)
01500	INIT ODP (OT III)
01510	INIT TOP (OT III)

Figure 3.4 Sample Standard Title Dataset Listing (cont'd)

01520	INIT ODP <OT III>
01530	REVISE CTP-III <DRAFT>
01540	FREQ ASSIGNMENT
01550	DEVA <IPR>
01560	DEVA <IPR> APPROVAL
01570	TYPE CLSFN <ED>
01580	PREP ECP <A KIT>
01590	APPR ECP <A KIT>
01600	PREP ECP <B KIT>
01610	APPR ECP <B KIT>
01620	PREP MWO <A KIT>
01630	APPR MWO <A KIT>
01640	PREP MWO <B KIT>
01650	APPR MWO <B KIT>
01660	** LRIP *****
01670	REVISE CTP-III
01680	DETMN AND FINDING <LRIP>
01700	SPECS <LRIP>
01710	STATEMENT OF WORK <LRIP>
01720	DATA CALL<LRIP>
01730	DATA REQMTS REVIEW BD <LRIP>
01740	CONTR DATA REQMTS LIST <LRIP>
01750	CONTR TEST REQMTS <LRIP>
01760	PROOBY ENGRG & PLANNING <LRIP>
01770	PREP EVAL CRITERIA <LRIP>
01780	PREP SEC CLSFN GUIDE <LRIP>
01790	INDPT GOVT COST EST <LRIP>
01800	SOLE SOURCE DECISION <LRIP>
01810	RFP <LRIP>
01820	RFQ <LRIP>

Figure 3.4 Sample Standard Title Dataset Listing (cont'd)

01830	EVAL CONTR PROPOSALS <LRIP>
01840	KO TEAM REVIEW <LRIP>
01850	CONTRACT AWARDS BD <LRIP>
01860	CONTRACT AWARD <LRIP>
01870	FINAL TDP <DT III>
01880	FINAL ODP <DT III>
01890	FINAL TDP <OT III>
01900	FINAL ODP <OT III>
01910	CTP-III <FINAL>
01920	MMO
01930	NET FOR INSTR KEY DEPOT PERS
01940	AIRCRAFT BAILED <LRIP>
01950	DT III <PUT-C>
01960	DT III <PUT-C> REPT<PRLM>
01970	DT III <PUT-C> REPT<REVIEW>
01980	DT III <PUT-C> REPT<FINAL>
01990	SAFETY OF FLT RELEASE <LRIP>
02000	DT III <PUT-G>
02010	DT III <PUT-G> REPT<PRLM>
02020	DT III <PUT-G> REPT<REVIEW>
02030	DT III <PUT-G> REPT<FINAL>
02040	DT III <DTP>
02050	DT III <DTP> REPT<PRLM>
02060	DT III <DTP> REPT<REVIEW>
02070	DT III <DTP> REPT<FINAL>
02080	DT III INOPT EVAL
02090	DT III INOPT EVAL
02100	DT III INOPT EVAL <REPT>
02110	DT III INOPT EVAL <REPT>
02120	UPDATE TECH DATA PKG

Figure 3.4 Sample Standard Title Dataset Listing (cont'd)

02130	UPDATE PLAN FOR LOG SUPT
02140	PREP BOIP II
02150	APPR BOIP II
02160	FINAL GQPRI AND MOS DECISION
02170	PHYS CONF AUDIT(PCA)
02180	UPDATE TOP
02190	PU (IPR)
02200	PU (IPR) APPROVAL
02210	** FULL PROO *****
02220	TYPE CLSFN (P)
02230	FIRST EDITION TM
02240	NET FOR TNG INSTRUCTORS
02250	INSTL OF TNG DEVICES & EQPT
02260	DEPOT PARTS(KITS) FILL
02270	NEW EQPT INTRO TEAM:
02280	NET TNG OF USERS
02290	INSTL OF MODS
02300	DETMN AND FINDING (P)
02320	STATEMENT OF WORK (P)
02330	DATA CALL(P)
02340	DATA REQMTS REVIEW BD (P)
02350	CONTR DATA REQMTS LIST (P)
02360	CONTR TEST REQMTS (P)
02370	PROOBY ENGRG & PLANNING (P)
02380	PREP EVAL CRITERIA (P)
02390	PREP SEC CLSFN GUIDE (P)
02400	INDPT GOVT COST EST (P)
02410	SOLE SOURCE DECISION (P)
02420	RFP (P)
02430	RFQ (P)

Figure 3.4 Sample Standard Title Dataset Listing (cont'd)

02440	EVAL CONTR PROPOSALS (P)
02450	KO TEAM REVIEW (P)
02460	CONTRACT AWARDS BD (P)
02470	CONTRACT AWARD (P)
02480	PREPARATION FIELD INTRO
02490	EXECUTION FIELD INTRO
02500	INIT OPER CAPABILITY (IOC)
READY	

Figure 3.4 Sample Standard Title Dataset Listing (cont'd)

DO YOU NEED FULL PROMPTING ?

Y
ENTER SYSTEM TYPE

N - NEW (THE SYSTEM DOES NOT HAVE A CODE)
O - OLD (THE SYSTEM IS IN THE MAIN FILE)

N
WILL YOU ENTER THE DATA FROM THE KEYBOARD OR IS IT PRESENTLY ON A FILE ?

ENTER

K - KEYBOARD

F - FILE

N - NONE

K

Full Prompting

DO YOU NEED FULL PROMPTING ?

N

ENTER SYSTEM TYPE

N

WILL YOU ENTER THE DATA FROM THE KEYBOARD OR IS IT PRESENTLY ON A FILE ?

K

No Prompting

Figure 4.1 Full/No Prompting Examples

ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS

TASK OR EVENT

FOLLOWED BY NUMBER

E / ID# / EVENT / DATE /
/ * / THIS IS A SINGLE EVENT / 1 * JUN * 79 /

SINGLE EVENT

ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS

TASK OR EVENT

FOLLOWED BY NUMBER

E2 / ID# / EVENT / DATE /
/ * / THIS IS A MULTIPLE EVENT / 1 * JUN * 80 /
/ EVENT DATE /
/ 1 * DEC * 81 /

MULTIPLE EVENT

Figure 4.2 Sample Single and Multiple Event Input

SYSTEM NUMBER: 001
 MAJOR SYSTEM: DEV COMMAND BUSINESS SYSTEMS
 PROJECT TITLE: SCHEDULE
 AS OF DATE: 12 AUG 77
 0010 (E) UP TO 9 EVENTS PER LINE

0020	(E) UP TO 3 TASKS PER LINE	08XJANX77 17XFEBX78 100
		25XMARX84 04XAPRX79 030
		14XMAYX80 28XJUNX81 001
		01XJULX82 21XAUGX83
		31XAUGX80
		03XJANX77 24XNOVX77 100
		14XMARX79 31XDECX81 030
		01XMAYX82 21XJUNX83 001
0030	(E) TEST FOR 1 JAN 78 & 1 JAN 79	31XDECX77 01XJANX79 060
0040	(E) TEST 0% COMPLETE	11XFEBX78 04XJANX79 050
0050	(E) TEST 50% AND 100% COMPLETE	12XFEBX78 01XJANX79 100
		16XJANX80 31XMARX80 005
0060	(E) A LONG TASK(5% COMPLETE)	13XJANX77 21XFEBX83 100
0070	(E) A SHORT TASK(100% COMPLETE)	01XMAYX77 30XJUNX77
0080	(E) THIS IS A 31 DEC 76 EVENT	31XDECX76
0090	(E) CHECK FOR X COMPLETE	31XDECX75
0110	(E) CHECK FOR X COMPLETE	09XJANX77 13XDECX80 060
0120	(E) THIS INTERACTIVE PROGRAM WAS DEVELOPED FOR THE ASE PMO BY PERSONNEL IN	
0130	(E) THE AURADCOM D&E DIRECTORATE. IT	
0140	(E) PERMITS THE USER TO ENTER TASKS AND/OR	
0150	(E) EVENTS WITH APPROPRIATE DATES FROM	
	(E) A TERMINAL AT HIS DESK. THE SCHEDULE	
0160	(E) CHART IS THEN PRODUCED ON A TEKTRONIX	
	(E) 4014. UPDATE OF THE DATA IS EASILY	
0170	(E) ACCOMPLISHED AT THE USER'S DISCRETION.	
	(E) "SCHEDULE" IS OPERATIONAL ON THE	
	(E) MIDWEST SCIENTIFIC & ENGINEERING	
	(E) COMPUTER.	

Figure 4.4 Sample Graphic Report Record Listing

ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS
TASK OR EVENT
FOLLOWED BY NUMBER

T TITLE / STR.DT. / STP.DT. / COMPLT. /
/SINGLE TASK ADDITION / 1*JAN*80 / 1*DEC*80 / ***_0_*** /
APPLICATION CODE
/SET /

SINGLE TASK ADDITION

ENTER ELEMENT TYPE AND NUMBER OF ELEMENTS
TASK OR EVENT
FOLLOWED BY NUMBER

T 2 TITLE / STR.DT. / STP.DT. / COMPLT. /
/MULTIPLE TASK ADDITION / 1*DEC*80 / 15*JUN*81 / ***_0_*** /
/STR.DT. / STP.DT. / COMPLT. /
/15*JUN*81 / 1*JUL*81 / ***_0_*** /
APPLICATION CODE
/SET /

MULTIPLE TASK ADDITION

Figure 4.5 Sample Single and Multiple Task Input


```

SYSTEM NUMBER: 001
MAJOR SYSTEM: DEV COMMAND BUSINESS SYSTEMS
PROJECT TITLE: SCHEDULE
AS OF DATE: 12 AUG 77
0010 (E) UP TO 9 EVENTS PER LINE

0020 (E) UP TO 3 TASKS PER LINE

0030 (E) TEST FOR 1 JAN 78 & 1 JAN 79
0035 (SET) ADD EXAMPLE
0040 (E) TEST 0% COMPLETE
0050 (E) TEST 50% AND 100% COMPLETE

0060 (E) A LONG TASK(5% COMPLETE)
0070 (E) A SHORT TASK(100% COMPLETE)
0080 (E) THIS IS A 31 DEC 76 EVENT
0090 (E) CHECK FOR X COMPLETE
0110 (E) CHECK FOR X COMPLETE
0120 (E) THIS INTERACTIVE PROGRAM WAS DEVELOPED
      FOR THE ASE PMO BY PERSONNEL IN
0130 (E) THE AURADCOM D&E DIRECTORATE. IT
      PERMITS THE USER TO ENTER TASKS AND/OR
0140 (E) EVENTS WITH APPROPRIATE DATES FROM
      A TERMINAL AT HIS DESK. THE SCHEDULE
0150 (E) CHART IS THEN PRODUCED ON A TEKTRONIX
      4014. UPDATE OF THE DATA IS EASILY
0160 (E) ACCOMPLISHED AT THE USER'S DISCRETION.
      'SCHEDULE' IS OPERATIONAL ON THE
0170 (E) MIDWEST SCIENTIFIC & ENGINEERING

      08XJANX77 17XFEBX78 100
      25XMARX84 04XAPRX79 030
      14XMAYX80 28XJUNX81 001
      01XJULX82 21XAUGX83
      31XAUGX80
      03XJANX77 24XNOVX77 100
      14XMARX79 31XDECX81 030
      01XMAYX82 21XJUNX83 001
      31XDECX77 01XJANX79
      01XJANX80
      11XFEBX78 04XJANX79 000
      12XFEBX78 01XJANX79 050
      16XJANX80 31XMARX80 100
      13XJANX77 21XFEBX83 005
      01XMAYX77 30XJUNX77 100
      31XDECX76
      31XDECX75 09XJANX78 060
      09XJANX77 13XDECX80 060

```

Figure 4.6 System Listing With New Record Added

SYSTEM NUMBER: 004			
MAJOR SYSTEM: DEV COMMAND BUSINESS SYSTEMS			
PROJECT TITLE: SCHEDULE			
AS OF DATE: 12 AUG 77			
0010	(E) UP TO 9 EVENTS PER LINE	08XJANX77 17XFEBX78	100
		25XMARX84 04XAPRX79	030
		14XMAYX80 28XJUNX81	001
		01XJULX82 21XAUGX83	
		31XAUGX80	
0020	(E) UP TO 3 TASKS PER LINE	03XJANX77 24XNOVX77	100
		14XMARX79 31XDECX81	030
		21XMAYX82 31XMAYX83	001
		01XJANX80	
		01XFEBX80	
0021	(SET) LIST TEST EVENT 1	01XJANX79	000
0022	(SET) LIST TEST EVENT 2	11XDECX77 04XJANX79	050
0030	(E) TEST FOR 1 JAN 78 & 1 JAN 79	12XFEBX78 01XJANX79	100
0040	(E) TEST 0X COMPLETE	16XJANX80 31XMARX80	005
0050	(E) TEST 50X AND 100X COMPLETE	13XJANX77 30XJUNX77	100
		01XMAYX77	
0060	(E) A LONG TASK(5X COMPLETE)	31XDECX76	060
0070	(E) A SHORT TASK(100X COMPLETE)	31XDECX75	060
0080	(E) THIS IS A 31 DEC 76 EVENT	09XJANX77 13XDECX80	
0090	(E) CHECK FOR X COMPLETE		
0100	(E) CHECK FOR X COMPLETE		
0110	(E) THIS INTERACTIVE PROGRAM WAS DEVELOPED FOR THE ASE PMO BY PERSONNEL IN		
0120	(E) THE AURADCOM D&E DIRECTORATE. IT		
0130	(E) PERMITS THE USER TO ENTER TASKS AND/OR		
0140	(E) EVENTS WITH APPROPRIATE DATES FROM		
0150	(E) A TERMINAL AT HIS DESK. THE SCHEDULE		
	(E) CHART IS THEN PRODUCED ON A TEKTRONIX		
	(E) 4014. UPDATE OF THE DATA IS EASILY		
	(E) ACCOMPLISHED AT THE USER'S DISCRETION.		
	(E) 'SCHEDULE' IS OPERATIONAL ON THE		
0160	(E) MIDWEST SCIENTIFIC & ENGINEERING		
	COMPUTER.		
ENTER EDIT			

Figure 4.7 Sample System Listing

SYSTEM NUMBER: 004
MAJOR SYSTEM: DEV COMMAND BUSINESS SYSTEMS
PROJECT TITLE: SCHEDULE
AS OF DATE: 12 AUG 77
0021 (SET) LIST TEST EVENT 1
0022 (SET) LIST TEST EVENT 2
END OF MAIN FILE

01XJANX80
01XFEBX80

Figure 4.8 Sample Output Listing

```

SYSTEM NUMBER: 004
MAJOR SYSTEM: DEV COMMAND BUSINESS SYSTEMS
PROJECT TITLE: SCHEDULE
AS OF DATE: 12 AUG 77
0010 (E) UP TO 9 EVENTS PER LINE

0020 (E) UP TO 3 TASKS PER LINE

0021 (SET) LIST TEST EVENT 1
0022 (SET) LIST TEST EVENT 2
0030 (E) TEST FOR 1 JAN 78 & 1 JAN 79
0040 (E) TEST 0X COMPLETE
0050 (E) TEST 50X AND 100X COMPLETE

0060 (E) A LONG TASK(5X COMPLETE)
0070 (E) A SHORT TASK(100X COMPLETE)
0080 (E) THIS IS A 31 DEC 76 EVENT
0090 (E) CHECK FOR X COMPLETE
0100 (E) CHECK FOR X COMPLETE
0110 (E) THIS INTERACTIVE PROGRAM WAS DEVELOPED
    FOR THE ASE PMO BY PERSONNEL IN
    (E) THE AURADCOM D&E DIRECTORATE. IT
    PERMITS THE USER TO ENTER TASKS AND/OR
    (E) EVENTS WITH APPROPRIATE DATES FROM
    A TERMINAL AT HIS DESK. THE SCHEDULE
    (E) CHART IS THEN PRODUCED ON A TEKTRONIX
    4014. UPDATE OF THE DATA IS EASILY
    (E) ACCOMPLISHED AT THE USER'S DISCRETION.
    "SCHEDULE" IS OPERATIONAL ON THE

0120 (E) MIDWEST SCIENTIFIC & ENGINEERING
0130 COMPUTER.
0140
0150
0160

```

```

08XJANX77 17XFEBX78
25XMARX84 04XAPRX79
14XMAYX80 28XJUNX81
01XJULX82 21XAUGX83
31XAUGX80
03XJANX77 24XNOVX77
14XMARX79 31XDECX81
21XMAYX82 31XMAYX83
01XJANX80
01XFEBX80
31XDECX77
11XFEBX78
12XFEBX78
16XJANX80
13XJANX77
01XMAYX77
31XDECX76
31XDECX75
09XJANX77
09XJANX78
13XDECX80

```

```

100
030
001
000
050
100
005
100
060
060

```

Figure 4.9 Sample Output Listing

SAMPLE REPORT (NOT SLIPPED)															PAGE 1 OF 1		
PREPARED BY- SECO		MAJOR SYSTEM- DEU COMMAND BUSINESS SYSTEMS				TITLE- SCHEDULE				AS OF DATE 12 AUG 77							
		FY 77	FY 78	FY 79	FY 80	FY 81	FY 82	FY 83	FY 84								
		2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1
		CV 77	CV 78	CV 79	CV 80	CV 81	CV 82	CV 83	CV 84								
TASKS/EVENTS																	
1	UP TO 9 EVENTS PER LINE	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
2	UP TO 3 TASKS PER LINE	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
3	TEST FOR 1 JAN 78 & 1 JAN 79	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
4	TEST 0% COMPLETE	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
5	TEST 50% AND 100% COMPLETE	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
6	A LONG TASK(5% COMPLETE)	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
7	A SHORT TASK(100% COMPLETE)	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
8	THIS IS A 31 DEC 76 EVENT	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
9	CHECK FOR % COMPLETE	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
10	CHECK FOR % COMPLETE	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
REMARKS-		<p>THIS INTERACTIVE PROGRAM WAS DEVELOPED FOR THE USE PMO BY PERSONNEL IN THE AURADCOM DAE DIRECTORATE. IT PERMITS THE USER TO ENTER TASKS AND/OR EVENTS WITH APPROPRIATE DATES FROM A TERMINAL AT HIS DESK. THE SCHEDULE CHART IS THEN PRODUCED ON A TELETYPE 4014 UPDATE OF THE DATA IS EASILY ACCOMPLISHED AT THE USER'S DISCRETION. THE SCHEDULE IS OPERATIONAL ON THE MIDWEST SCIENTIFIC & ENGINEERING COMPUTER.</p>															

Figure 4.10 Sample Report (Not Slipped)

SAMPLE REPORT (SLIPPED)															PAGE 1 OF 1		
PREPARED BY:-		MAJOR SYSTEM- DEV COMMAND				TITLE- SCHEDULE										AS OF DATE	
SECO		BUSINESS SYSTEMS														12 AUG 77	
		FV 77	FV 78	FV 79	FV 80	FV 81	FV 82	FV 83	FV 84								
		2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1
		CV 77	CV 78	CV 79	CV 80	CV 81	CV 82	CV 83	CV 84								
L	TASKS/EVENTS	J F M A M J J A S O N D J F M A M J J A S O N D J F M A M J J A S O N D J F M A M J J A S O N D															
1	UP TO 9 EVENTS PER LINE	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
2	UP TO 3 TASKS PER LINE	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
3	TEST FOR 1 JAN 78 & 1 JAN 79	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
4	TEST OR COMPLETE																
5	TEST 50% AND 100% COMPLETE																
6	A LONG TASK (5% COMPLETE)	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
7	A SHORT TASK (100% COMPLETE)	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
8	THIS IS A 31 DEC 76 EVENT	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
9	CHECK FOR % COMPLETE	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
10	CHECK FOR % COMPLETE	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
REMARKS-		THIS INTERACTIVE PROGRAM WAS DEVELOPED FOR THE USE PRO BY PERSONNEL IN THE ALPACOM DAE DIRECTORATE. IT PERMITS THE USER TO ENTER TASKS AND/OR EVENTS WITH APPROPRIATE DATES FROM A TERMINAL AT HIS DESK. THE SCHEDULE CHART IS THEN PRODUCED ON A TELETYPE IX 401. UPDATE OF THE DATA IS EASILY ACCOMPLISHED AT THE USER'S DISCRETION. SCHEDULE IS OPERATIONAL ON THE MIDWEST SCIENTIFIC & ENGINEERING COMPUTER.															

Figure 4.11 Sample Report (Slipped)

SAMPLE REPORT (GRAPHICS)																					
	PREPARED BY:- SECO	MAJOR SYSTEM- DEV COMMAND BUSINESS SYSTEMS				TITLE-	SCHEDULE		PAGE 1 OF 1												
		FV 77	FV 78	FV 79	FV 80	FV 81	FV 82	FV 83	FV 84												
		CY 77	CY 78	CY 79	CY 80	CY 81	CY 82	CY 83	CY 84												
L I N E		2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1
TASKS/EVENTS																					
1 UP TO 9 EVENTS PER LINE																					
2 UP TO 3 TASKS PER LINE																					
3 TEST FOR 1 JAN 78 & 1 JAN 79																					
4 TEST OR COMPLETE																					
5 TEST 50% AND 100% COMPLETE																					
6 A LONG TASK(5% COMPLETE)																					
7 A SHORT TASK(100% COMPLETE)																					
8 THIS IS A 31 DEC 76 EVENT																					
9 CHECK FOR % COMPLETE																					
10 CHECK FOR X COMPLETE																					
REMARKS-	THIS INTERACTIVE PROGRAM WAS DEVELOPED FOR THE USE WHO BY PERSONNEL IN THE AUSTRACOM DAE DIRECTORATE. IT PERMITS THE USER TO ENTER TASKS AND/OR EVENTS WITH APPROPRIATE DATA FROM A TERMINAL AT HIS DESK. THE SCHEDULE CHART IS THEN PRODUCED ON A TELETYPE UNIT. AN UPDATE OF THE DATA IS EASILY ACCOMPISHED AT THE USER'S DISCRETION. THE SCHEDULE IS OPERATIONAL ON THE MIDWEST SCIENTIFIC & ENGINEERING COMPUTER.																				

Figure 4.12 Sample Graphic Report

SAMPLE REPORT (PRINTER)			
CURRENT DATE: 06*JUL*78		PAGE 1 OF 1	
PREPARED BY:SECO			
MAJOR SYSTEM: DEV COMMAND BUSINESS SYSTEMS			
PROJECT TITLE: SCHEDULE			
AS OF DATE: 12 AUG 77			
TASK/EVENT	START/EVENT DATE	STOP/EVENT DATE	STATUS
1. UP TO 9 EVENTS PER LINE	08*JAN*77	17*FEB*78	
	25*MAR*84	04*APR*79	
	14*MAY*80	28*JUN*81	
	01*JUL*82	21*AUG*83	
	31*AUG*80		
2. UP TO 3 TASKS PER LINE	03*JAN*77	24*NOV*77	100
	14*MAR*79	31*DEC*81	030
	01*MAY*82	21*JUN*83	001
	31*DEC*77	01*JAN*79	
3. TEST FOR 1 JAN 78 & 1 JAN 79	11*FEB*78	04*JAN*79	000
4. TEST 0% COMPLETE	12*FEB*78	01*JAN*79	050
5. TEST 50% AND 100% COMPLETE	16*JAN*80	31*MAR*80	100
	13*JAN*77	21*FEB*83	005
6. A LONG TASK(5% COMPLETE)	01*MAY*77	30*JUN*77	100
7. A SHORT TASK(100% COMPLETE)			
8. THIS IS A 31 DEC 76 EVENT			
9. CHECK FOR % COMPLETE	09*JAN*77	09*JAN*78	060
10. CHECK FOR % COMPLETE		13*DEC*80	060

THIS INTERACTIVE PROGRAM WAS DEVELOPED FOR THE ASE PMO BY PERSONNEL IN THE AVRADCOM D&E DIRECTORATE. IT PERMITS THE USER TO ENTER TASKS AND/OR EVENTS WITH APPROPRIATE DATES FROM A TERMINAL AT HIS DESK. THE SCHEDULE CHART IS THEN PRODUCED ON A TEKTRONIX 4014. UPDATE OF THE DATA IS EASILY ACCOMPLISHED AT THE USER'S DISCRETION. "SCHEDULE" IS OPERATIONAL ON THE MIDWEST SCIENTIFIC & ENGINEERING COMPUTER.

Figure 4.13 Sample Typewriter Report

PAGE 1 OF 1

PREPARED BY-

0035

**MAJOR SYSTEMS- DEV COMMAND
BUSINESS SYSTEMS**

7741- SCHEDULE

DATE

TASKS/EVENTS

UP TO 9 EVENTS PER LINE

UP TO 3 TASKS PER LINE

TEST FOR 1 JAN 78 & 1 JAN 79

TEST ON COMPLETE

TEST 50% AND 100% COMPLETE

A LONG TASK (5% COMPLETE)

A SHORT TASK (100% COMPLETE)

THIS IS A 31 DEC 78 EVENT

CHECK FOR COMPLETE

CHECK FOR A COMPLETE

REMARKS-

THIS INTERACTIVE PROGRAM WAS DEVELOPED FOR THE USE PRO BY PERSONNEL IN THE AUSA/COM IAW DISCREETLY. IT PERMITS THE USER TO ENTER TASKS AND/OR EVENTS WITH APPROXIMATE DATE, FROM A TERMINAL AT HIS DESK. THE SCHEDULE SHEETS IN THE PROGRAM ARE ON A SELECTIONX 4014. UPDATE OF THE DATA IS FULLY ACCOMPLISHED AT THE USER'S DISCRETION. SCHEDULE IS OPERATIONAL ON THE MIDWEST SCIENTIFIC & ENGINEERING COMPUTER.

Figure 4.14 Sample Report (No Window)

SAMPLE REPORT (WINDOWED)														PAGE 1 OF 1	
PREPARED BY- SECO		MAJOR SYSTEM- DEV COMMAND BUSINESS SYSTEMS				VITAL- SCHEDULE				AS OF DATE 12 AUG 77					
TASKS/EVENTS		PV		79		3		4		1					
		CV		79											
		J	F	M	A	M	J	J	A	S	O	N	D		
1	UP TO 9 EVENTS PER LINE														
2	UP TO 3 TASKS PER LINE														
3	TEST FOR 1 JAN 78 & 1 JAN 79														
4	TEST 80% COMPLETE														
5	TEST 50% AND 100% COMPLETE														
6	A LONG TASK (5% COMPLETE)														
7	CHECK FOR % COMPLETE														
REMARKS- THIS INTERACTIVE PROGRAM WAS DEVELOPED FOR THE USE AND BY PERSONNEL IN THE AUMACON HALL DIRECTORATE. IT PERMITS THE USER TO ENTER TASKS AND/OR EVENTS WITH ASSOCIATED DATES FROM A TERMINAL AT HIS DESK. THE SCHEDULE REPORT IS THEN PRODUCED ON A TELETYPE. *DISCONTINUATION OF DATA IS EARLYLY ACCOMPLISHED AT THE USER'S DISCRETION. SCHEDULE IS OPERATIONAL ON THE MIDWEST SCIENTIFIC & ENGINEERING COMPUTER.															

Figure 4.15 Sample Report (Windowed)

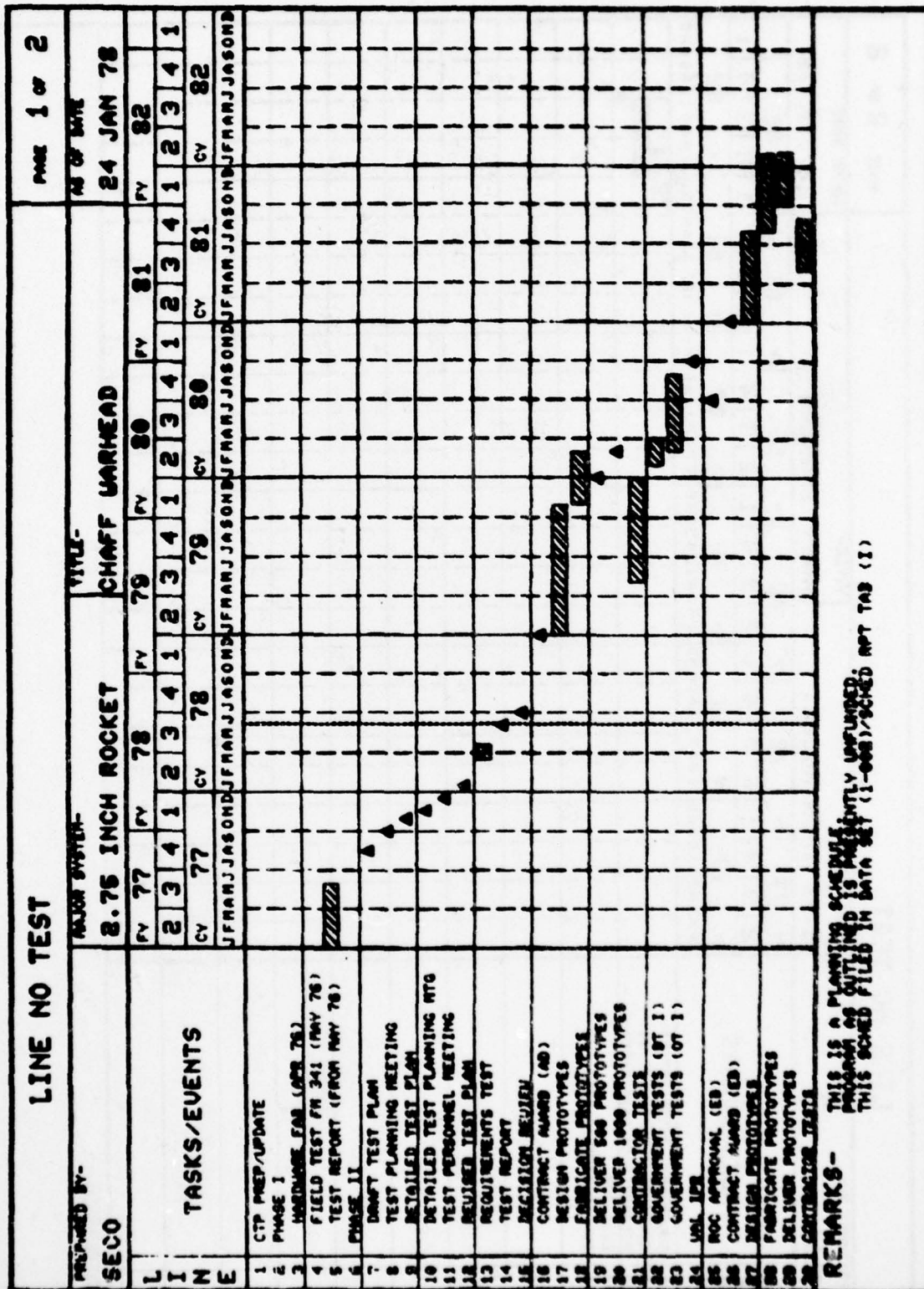


Figure 4.16 Lines Per Page Off

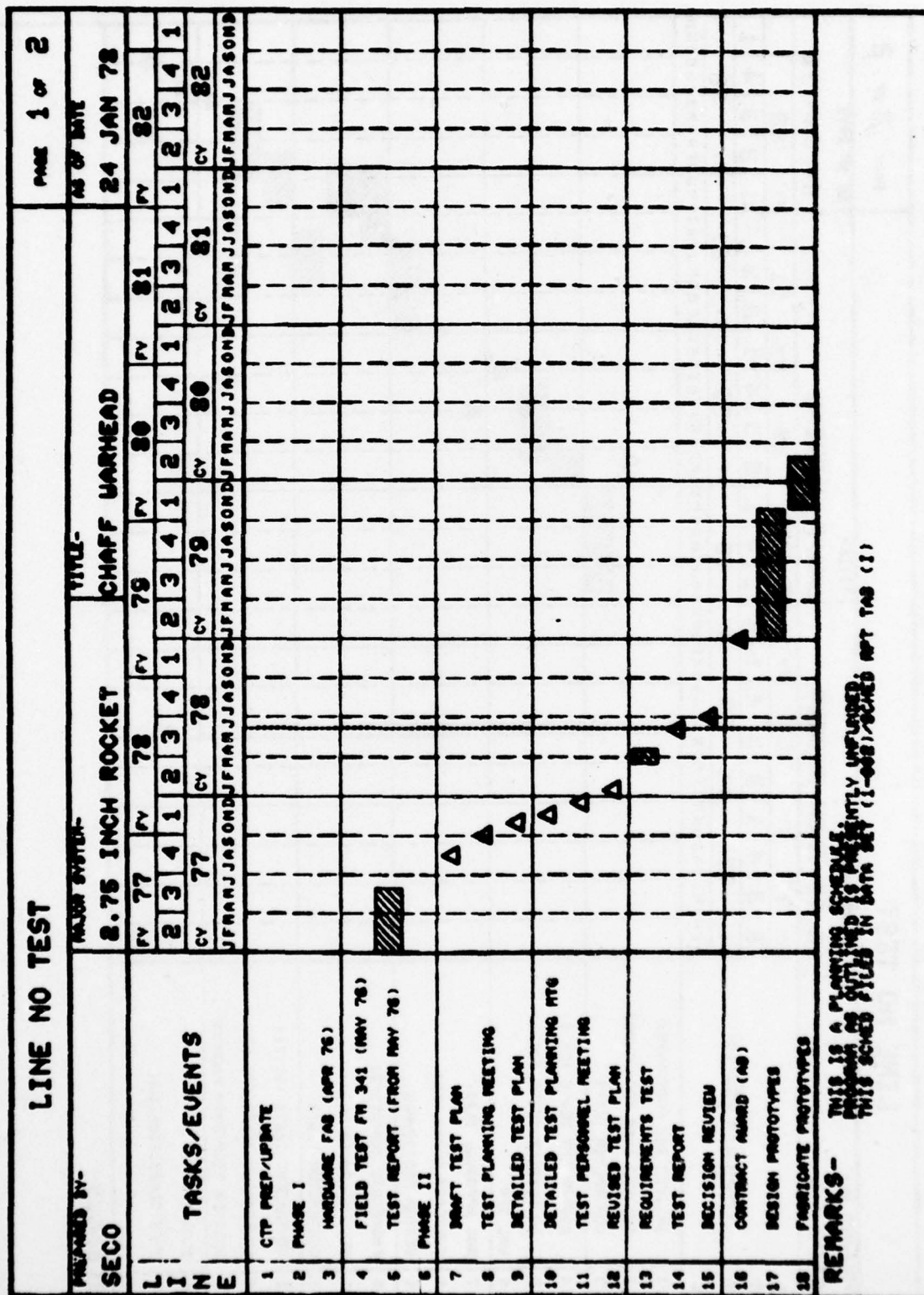


Figure 4.17 Lines Per Page On

DISTRIBUTION

Commander
US Army Materiel Development and Readiness Command
ATTN: DRCPM (COL Eek)
DRCMS-I
DRCDE-P
5001 Eisenhower Avenue
Alexandria, VA 22333

1
2
2

Commander
US Army Aviation Research and Development Command
ATTN: DRDAV-ES
DRDAV-EO
DRDAV-EX
DRDAV-B
DRDAV-Q (Allen)
DRDAV-N (Law)
PO Box 209
St. Louis, MO 63166

50
2
1
2
2
1

Commander
US Army Avionics Research and Development Activity
Fort Monmouth, NJ 07703

1

Commander
US Army Aviation Engineering Flight Activity
Edwards AFB, CA 93523

1

Commander
US Army Communications and Electronics
ATTN: DRSEL-MS
Fort Monmouth, NJ 07703

5

Commander
Automated Logistics Management Systems Agency
ATTN: DRXAL-FDB (Mayne)
210 N. 12th Street
St. Louis, MO 63101

3

AD-A072 140

ARMY AVIATION RESEARCH AND DEVELOPMENT COMMAND ST LO--ETC F/G 9/2
SCHEDULE. MANAGERIAL TOOL FOR PROJECT/PRODUCT MANAGERS.(U)
MAY 79 V ALLEN, L ANTWILER, M GARSIK

UNCLASSIFIED

USAAVRADCOM-TR-79-17

NL

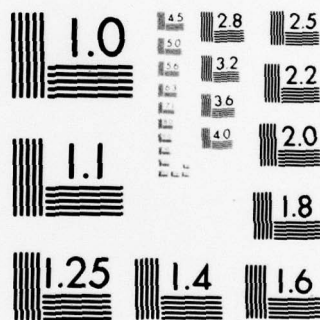
3 OF 3

AD
A072140



END
DATE
FILMED
9-79

DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Distribution Continued.

Project Manager
Remotely Piloted Vehicles
PO Box 209
St. Louis, MO 63166

1

Project Manager
Aircraft Survivability Equipment
PO Box 209
St. Louis, MO 63166

2

Project Manager
Black Hawk
PO Box 209
St. Louis, MO 63166

1

Project Manager
CH-47 Modernization Program
PO Box 209
St. Louis, MO 63166

1

Project Manager
Navigation/Control Systems
PO Box 209
St. Louis, MO 63166

1

Project Manager
Advanced Scout Helicopter
PO Box 209
St. Louis, MO 63166

1

Program Manager
Advanced Attack Helicopter
PO Box 209
St. Louis, MO 63166

1

Distribution Continued

**Director
US Army Research and Technology Laboratories
Ames Research Center
(Mail Stop 207-5)
Moffett Field, CA 94035**

1

**Director
Aeromechanics Laboratories
US Army RTL
Ames Research Center
(Mail Stop 215-1)
Moffett Field, CA 94035**

1

**Director
Propulsion Laboratory
US Army RTL
Lewis Research Center
(Mail Stop 77-5)
21000 Brookpark Road
Cleveland, OH 44135**

1

**Director
Structures Laboratory
NASA/Langley Research Center
(Mail Stop 26)
Hampton, VA 23665**

1

**Director
Applied Technology Laboratory
US Army RTL
Fort Eustis, VA 23604**

1

Distribution Continued

**US Army Plant Representative
Bell Helicopter - Textron
PO Box 1605
Fort Worth, TX 76101**

1

**US Army Plant Representative
Hughes Helicopter - Summa Corp.
2560 Walnut Avenue
Building 305T22
Venice, CA 90291**

1

**US Army Plant Representative
Boeing Vertol Company
PO Box 16859
Philadelphia, PA 19142**

1